and $j \in [0, N-1]$. Here, $\mathbb{1}_{[k \neq i]}$ is an indicator function evaluating to 1 for $k = i$, otherwise evaluating to 0. $\tau = 0.1$ denotes a temperature hyper-parameter.

SimCLR loss can be interpreted as a one hot classification problem—for each example, identifying its positive pair amongst all other (negative) examples in the batch. Because the batch-size determines the number of negative examples in the batch, and hence, the likelihood of non-trivial negatives, a large batch size of N pairs is crucial to this learning strategy, with larger batch sizes resulting in notable performance improvements [9]. However, batch sizes are limited to the memory of the available compute resources (e.g., of GPUs / TPUs). Hence, when considering compute costs, it is desirable to reduce batch size.

The primary objective of this work is to provide a comparative analysis of the utility of supervised and unsupervised learning strategies for a range of music understanding tasks under different data sources. As such we do not propose to investigate or innovate on the model architecture defining $f(\mathbf{X})$ itself, and employ the Short-Fast Normalizer-Free Net F0 (SF-NFNet-F0) of [2]. This architecture was chosen due to its demonstrated excellent performance in audio understanding, its design intent specifically for audio, and its use of efficient operations such as grouped convolutions. Furthermore, this architecture does not employ batch-normalization, improving training time and resource requirements, particularly when using SimCLR loss, by removing the need to globally synchronize data between GPU devices at each batch-normalization layer. Our intent was to reproduce this model as accurately as possible. However, due to the information available to us at the time of publication, there may be some discrepancies between our implementation and that in [2]. To ensure reproducibility, we release a Tensorflow implementation of the model used in this paper with SCOOCH configurations [1] for each of the models trained [2]. Our SF-NFNet-F0 implementation contains 62.4M parameters.

In both supervised and unsupervised settings we employ mixup [30] directly on the sampled log-mel spectrograms in real-time during training for data augmentation. We acknowledge that additively combined audio sources do not result in additively combined magnitude spectrograms due to both constructive and destructive interference. However, such an augmentation is an efficient operation to perform in real-time data sampling pipelines, mitigating data bandwidth bottlenecks that may be caused by more complex operations. We employ mixup by shuffling features within each batch and additively combining the shuffled features (and labels for supervised learning) to the original batch. Mixup gains are sampled from a beta distribution with parameters $\alpha = 5.0$ and $\beta = 2.0$, for each feature in a batch.

In all pre-training contexts, we use an Adam optimizer with a learning rate following a warm-up cosine decay schedule, first increasing to 0.0002 over 5k steps, then decreasing to 0.0 over 195k steps. While dense projectors, $h(\mathbf{z})$ are an inherent part of the unsupervised SimCLR learning approach, we find projectors to also be useful in the supervised setting, adding a non-linear transformation between the learned embeddings, $\mathbf{z}$, and the supervised labels, $\hat{\mathbf{y}} = h(f(\mathbf{X}))$. We notice such an approach has also been useful in computer vision [31]. Hence, in all contexts we employ a projector consisting of 3×4096 node hidden layers with ReLU activation. Supervised models were trained on 8 v100 GPUs taking approximately 30 hours, while unsupervised models were trained on 16 A100 GPUs taking approximately 80 hours.

[1] https://github.com/PandoraMedia/scooch

[2] https://github.com/PandoraMedia/music-audio-representations

2.2 Datasets

Pre-training datasets can have a significant impact on supervised and unsupervised model performance. In the supervised context, there has been evidence that models are less generalizable to unseen tasks [2], perhaps due to the information present in the supervised labels. In the unsupervised context, less investigation has been conducted into the effect of the content of pre-training datasets. In this context, the problem of sampling positive and non-trivial negative examples within a dataset is closely tied to the diversity of content in that dataset. Furthermore, when employing mixup as a data augmentation strategy, the content of the pre-training dataset also defines the additive noise that is applied to features during training.

We compare pre-training on two large datasets, namely **Musicset** and a version of **Audioset** [32]. The Audioset training set contains $\approx$ 1.7M distinct labelled content pieces, and Musicset $\approx$ 1.8M. Audioset consists of 10 second snippets of various audio sources: music, speech and environmental audio (4,791 hours, $\approx$ 602 GB of audio feature data). Musicset, in contrast, focuses solely on music; it consists only of labelled complete songs, each up to several minutes in length (117,497 hours, $\approx$ 14.7 TB of audio feature data). Musicset is, to our knowledge, the largest dataset of expert-annotated audio ever trained on. While it is not publicly available, we believe it valuable to report the results of models trained on this dataset to communicate the effectiveness of supervised learning at this scale.

For supervised learning, in addition to the features themselves, the labels differ. Each dataset's vocabulary is distinct but similar in size (527 labels for Audioset and 500 labels for Musicset), however, Figure 2 shows the label density and distribution of both datasets differs.

2.3 Feature Sampling

The features, $\mathbf{X}$, are log-magnitude log-mel spectrograms produced from waveforms sampled at 16 kHz. We use 96 HTK-log-mel spaced, power normalized, frequency bins with center frequences from 0 Hz to 8 kHz, analyzed with a window size of 25 ms, a Fourier transform size of 2048 bins and a hop size of 10 ms. We sample 3 s snippets of each content piece in real-time (during training) across the pre-training dataset, forming features, $\mathbf{X} \in \mathbb{R}^{96 \times 300}$. Sampling features in real-time from full-length tracks has the benefit of a high ratio of distinct features per static dataset size by reducing redundant (overlapping) data storage. For

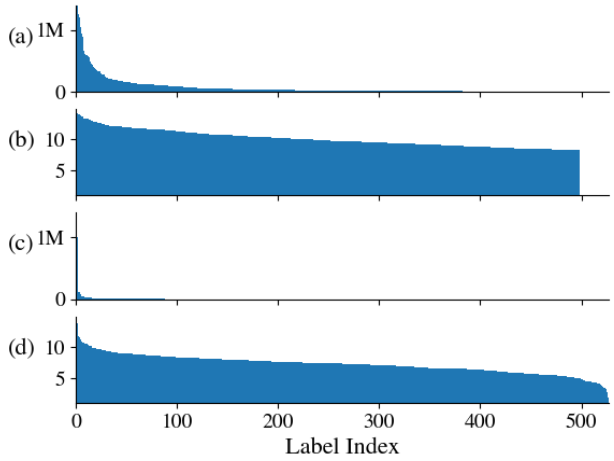

Figure 2. Label density distributions for Musicset and Audioset. (a), (b) plots the sorted Musicset label counts on a linear (in millions) and logarithmic scale, respectively. (c), (d) plots the same data for Audioset, respectively.

example, a 183 second song (6.8MB) results in 18k distinct features (each 112kB) with the above parameters.

To build a batch from a dataset, we first select a uniform random sample of tracks (with replacement); then, from each selected track, we choose a random 3-second spectrogram snippet, without padding. In the unsupervised context, we need to select positive and negative pairs for each selected snippet. Positive pairs are sampled from the same 10 second "track" for Audioset; for Musicset, positive examples are also sampled from the same track, although we require positive pairs to be centered on frames less than 5 seconds apart in the track's timeline. As negative examples, the SimCLR objective implicitly uses all other samples in a batch. We also tried forcing examples from the same track, but further away from the anchor than the positive, into every batch as hard negatives. Such an approach has shown promising results in music segmentation [33], however, we found that for music labelling at the averaged track-level this did not change our results significantly. As such, we omit this approach from our evaluation.

3. EVALUATION

To evaluate embedding performance, we take the global average pooled activations of the final convolutional layer in the SF-NFNet-F0 architecture for 3 second feature snippets sampled along the length of each audio timeline at a frequency of 0.5 Hz, then average these sampled embeddings along the length of the timeline (with the exception of NSynth datasets as described later in this section). We found similar results training probes on both the timeline-averaged embeddings, and individual embeddings sampled directly from track timelines, with slight improvements when using timeline-averaged embeddings in some cases.

Each of the embeddings derived from pre-trained models have different levels of predictive power and will underfit / overfit with different parameters for each downstream-dataset / pre-trained model combination. To demonstrate the potential of embeddings, and to investigate whether a given embedding can achieve SOTA performance on a given dataset, it is important to optimize the probe parameters for each dataset. However, in cases where such a transfer learning methodology is SOTA, it is important to constrain the probe to the limitations imposed in previous works so that it is the quality of embeddings that are evaluated, and not that of the probes. With these considerations, we adopt a hybrid strategy based on the prior SOTA for each dataset. We optimize the parameters of probes in all cases except where the prior SOTA adopts a similar transfer learning methodology, in which case, we constrain the probes to the parameter ranges investigated in those prior works. Regardless of prior works, we restrict all probes to simple multi-layer perceptron (MLP) classifiers which can be trained on a single 32 core CPU in less than 1 hour to maintain the aforementioned benefits of efficiency and scalability that such a transfer learning approach provides.

In all cases, we train probes using Adam optimization with a cosine learning rate schedule with 1,000 steps of warmup followed by a decay to zero over the remainder of the steps. We optimize the learning rate, number of training steps, and l2-regularization of each probe to achieve best performance / prevent overfitting.

In total we evaluate the performance over annotations of 7 distinct collections of audio, comprising 15 distinct datasets in total. An overview of datasets is shown in Table 1, and probe configurations in Table 2. We also publish more detailed SCOOCH configurations for probes, tag-level results, and model weights for the Musicset-ULarge model as a baseline for future research. [3] The remainder of this section covers specific notes on each dataset.

Dataset	Task	#Items	#Labels	Avg. secs	Hours
MSDS	tagging	242k	50	46	3.08k
MSD50	tagging	36k	50	46	464
MSD100	tagging	115k	100	47	1.50k
MSD500	tagging	156k	500	47	2.05k
AMM	mood	67k	188	45	840
MuMu	genre	147k	250	47	1.92k
MTT	tagging	26k	50	29	171
NSynth_P	pitch	306k	112	4	340
NSynth_I	instrument	306k	11	4	340
GTZAN	genre	930	10	30	7.8
Emo	emotion	744	N/A	45	9.3
GS_Key	key	2.1k	24	120	116
Jam-50	tagging	54k	50	244	3.69k
Jam-All	tagging	56k	183	244	3.76k
Jam-MT	mood/theme	18k	56	219	1.10k

Table 1. Overview of evaluation datasets. The sizes for MSD50 and MSD500 disagree with [22]. The published MSD50 splits include 35,745 IDs, and we exclude any tracks in MSD500 without any tags in the vocabulary.

Magnatagatune: Magnatagatune (MTT) [34] annotations include genre, instrumentation / vocal, mood, perceptual tempo, origin and sonority features. We adopt the common approach using the published splits and top 50 tags [4]. Some research removes tracks without tags in the most common 50 tags. To make results comparable to the

[3] https://github.com/PandoraMedia/music-audio-representations
[4] https://github.com/jongpillee/music_dataset_split

Dataset	Layers	Dropout	Batch (N)	Constraint
MSDS	2×1024	0.5	256	None
MSD50	1×512	0.5	256	None
MSD100	2×2048	0.5	256	None
MSD500	3×4096	0.5	256	None
AMM	1×1024	0.5	256	None
MuMu	linear	0.3	256	None
MTT	1×512	0.5	256	[24]
NSynth_P	linear	0.0	64	[2]
NSynth_I	linear	0.0	64	[2]
GTZAN	linear	0.0	2560	None
Emo	1×1024	0.5	512	None
GS_Key	1×512	0.8	512	None
Jam-50	1×512	0.75	256	None
Jam-All	2×1024	0.5	256	None
Jam-MT	1×512	0.75	256	None

Table 2. Probe parameters

most recent SOTA on this dataset [24], we keep items without tags in both training and evalution sets.

GTZAN: This dataset addresses the problem of genre classification in a single-label context [35]. We employ the fault-filtered version of this dataset [4].

NSynth: Here we evaluate detecting the pitch (NSynth_P) and instrument (NSynth_I) of individual musical notes. Following [2], in the case of NSynth_P we analyze the average of embeddings from four non-overlapping consecutive snippets of 1 second feature windows, and for NSynth_I we analyze a single embedding for each note produced using a 4 second feature window.

Million Song Dataset: The Million Song Dataset (MSD) contains labels pertaining to era, instrumentation, sonority, genre, mood, origin and activity. Because of the size of this dataset and variation in label quality throughout we adopt several different splits and vocabularies for the dataset. For comparability with the previous SOTA, we adopt the vocabulary of the 50 most common labels, and the splits employed in [15] [4], namely MSDS. We also take advantage of the several cleaned and artist-separated datasets available for this collection [22]—MSD50, MSD100 and MSD500, for which the splits and vocabularies are available publicly [5].

All Music Moods: The All Music Moods dataset (AMM) [36], focuses on mood prediction for the MSD audio data. Because the mood labels are heavily correlated with the artists in the dataset, we find it important to adopt the artist-seperated split [6]. With no previously published result on this split, we additionally provide results for embeddings produced by the MusiCNN model [37].

Jamendo: This dataset contains genre, instrument and mood / theme tags for audio from Jamendo [38]. We evaluate on all tags (Jam-All) and the 50 most common (Jam-50). Because very few of the mood / theme tags are in Jam-50, we also evaluate the mood / theme category (Jam-MT). We use the official splits [7] and full-length audio.

GiantSteps Key: This dataset (GS_Key) concerns major/minor key classification in electronic music—a 24-way classification problem. It combines two datasets: the first, 604 2-minute samples of electronic music tracks collected from beatport.com [39]; the second [8], consists of 1486 tracks from the same source. Consistent with previous work [23, 40], we use the former 604 samples for testing, and the latter for training / validation, selecting high-confidence annotations partitioned into train and validation sets according to [23]. For evaluation we compute a weighted accuracy score common in key classification [9].

EmoMusic: This dataset (Emo) concerns emotion recognition [41], and provides continuous annotations for valence and arousal. Following [23], we consider the static version of this dataset, where target values are averaged for each clip, and use the artist-based train / test partitioning provided in their work. This poses a regression problem—we use the coefficient of determination as the evaluation metric for both valence (Emo_V) and arousal (Emo_A).

MuMu: The Multimodal Music dataset (MuMu) [42], focuses on multilabel genre predictions for the MSD audio data, with genre annotations from the Amazon Reviews dataset. We evaluate using the official splits [10].

4. RESULTS

In order to derive conclusions on the potential performance of supervised models trained using binary labels on music data, we train a supervised model on the complete Musicset dataset, namely *Musicset-Sup*. To compare these results to supervised learning on a large-scale public dataset covering music, environmental, and speech audio, we provide results for a model trained on ≈1.7M items from Audioset, namely *Audioset-Sup*. To compare supervised and unsupervised learning at a similar scale, we provide results for a model trained using the unsupervised methodology discussed in Section 2 on the same Musicset dataset audio data and the Audioset audio data, namely *Musicset-ULarge* and *Audioset-ULarge* using a batch size of 1920 pairs.

In addition, we train two further unsupervised models using a batch-size of 256. One trained on Audioset (*Audioset-USmall*), and one on ≈1.8M randomly sampled 10 second snippets from Musicset, one per content piece (*Musicset-USmall*). The reasoning for this is two-fold—firstly by sampling short snippets from Musicset we mitigate the confounding variable of dataset size when comparing the two, secondly this allows us to investigate the effect of batch size in the unsupervised learning paradigm.

For convenience, we also include results for the recent evaluations in [2, 24], and those for any other previous SOTA, excluding these two evaluations. The results for each of these models is shown in Table 3 for all datasets excluding annotations of MSD, which are shown in Table 4. There, we note that for all multilabel music tagging tasks, the Musicset-Sup model achieves SOTA performance by significant margins. This is encouraging given that the Musicset training dataset was created naively, and the supervised information therein opens the door to improvements such as label-balanced sampling which has been shown to

[5] https://github.com/minzwon/tag-based-music-retrieval
[6] https://github.com/fdlm/listening-moods
[7] https://github.com/MTG/mtg-jamendo-dataset
[8] https://github.com/GiantSteps/giantsteps-mtg-key-dataset
[9] https://www.music-ir.org/mirex/wiki/2021:Audio_Key_Detection
[10] https://zenodo.org/record/1236906#.YoPIAhNBx0s

Model	MTT		GTZAN	$NSynth_P$	$NSynth_I$	Emo_V	Emo_A	GS_{Key}	Jam-50		Jam-All	
	mAP	ROC	Acc	Acc	Acc	r^2	r^2	W. Acc	mAP	ROC	mAP	ROC
Musicset-Sup	**0.413**	**0.917**	**0.835**	0.793	0.731	0.566	**0.726**	0.286	**0.321**	**0.843**	**0.162**	**0.839**
Audioset-Sup	0.386	0.904	0.748	0.819	0.676	0.341	0.545	0.210	0.284	0.822	0.135	0.813
Musicset-ULarge	**0.404**	**0.914**	0.735	**0.892**	0.740	0.577	0.700	0.667	**0.317**	**0.839**	**0.159**	**0.833**
Audioset-ULarge	0.391	0.906	0.672	0.805	0.721	0.438	0.624	0.287	0.285	0.826	0.131	0.816
Musicset-USmall	0.389	0.905	0.686	0.824	0.714	0.389	0.668	0.508	0.292	0.828	0.138	0.817
Audioset-USmall	0.375	0.897	0.648	0.777	0.698	0.386	0.609	0.197	0.268	0.817	0.127	0.809
Jukebox [23]	**0.414**	**0.915**	0.797	-	-	**0.617**	**0.721**	0.667	-	-	-	-
Prev. SF-NFNet-F0 [2]	0.395	-	-	0.880	**0.782**	-	-	-	-	-	-	-
SOTA Excl. [2, 23]	0.384 [37]	**0.92** [12]	0.821 [11]	- -	0.741 [43]	0.556 [44]	0.704 [45]	**0.796*** [46]	0.298 [47]	0.832 [47]	- -	- -

Table 3. Results for models on all datasets, excluding MSD annotations and Jam-MT. Models within 0.01 of SOTA are bold. *The SOTA for GS_{Key} has no publicly available implementation details and is specialized for this dataset, e.g., [48].

Model	MSDS		MSD50		MSD100		MSD500		MuMu		AMM		Jam-MT	
	mAP	ROC	mAP	ROC	mAP	ROC	mAP	ROC	mAP	ROC	mAP	ROC	mAP	ROC
Musicset-Sup	**0.363**	**0.903**	**0.459**	**0.913**	**0.346**	**0.906**	**0.169**	**0.898**	**0.257**	**0.908**	**0.180**	**0.791**	**0.161**	**0.786**
Audioset-Sup	0.308	0.880	0.375	0.883	0.278	0.877	0.128	0.874	0.191	0.867	0.156	0.760	0.137	0.749
Musicset-ULarge	0.351	**0.900**	0.438	**0.908**	0.321	**0.897**	0.152	**0.891**	0.235	0.893	**0.174**	**0.784**	**0.158**	**0.781**
Audioset-ULarge	0.311	0.885	0.377	0.886	0.276	0.878	0.121	0.873	0.162	0.855	0.156	0.763	0.142	0.765
Musicset-USmall	0.319	0.888	0.384	0.892	0.283	0.881	0.129	0.878	0.190	0.871	0.155	0.762	0.138	0.757
Audioset-USmall	0.286	0.876	0.353	0.878	0.251	0.870	0.110	0.868	0.152	0.850	0.151	0.753	0.136	0.753
SOTA	0.348 [15]	**0.897** [15]	0.386 [14]	**0.921** [14]	0.185 [22]	- -	- -	- -	- -	0.888* [42]	0.163 [37]	0.773 [37]	**0.161**† [49]	**0.781**† [49]

Table 4. Results for all models on MSD annotations and Jam-MTT. Models within 0.01 of SOTA are bold. *The previous SOTA for MuMu evaluated artist-level predictions rather than track-level. †The SOTA for Jam-MT is trained on an expanded within-taxonomy set ($\approx$ 16k tracks), here we restrict probe training to Jamendo tracks ($\approx$ 10k).

realize further performance gains [6]. We leave this investigation for future work.

It is also encouraging to see that unsupervised learning on music audio (Musicset-ULarge) achieves SOTA performance compared to previous unsupervised models, in addition to some supervised models (specifically, for models trained on Jam-50 and all MSD datasets).

Comparing to recent transfer learning approaches [2, 24], we see that in all cases either Musicset-Sup or Musicset-ULarge outperform or are on par with these, with the exception of $NSynth_I$. This is a promising result espousing the value of music only audio datasets considering that previously reported results for unsupervised learning on Audioset [2] used more than double the batch size of those in this paper. Wrt. the Jukebox model, models trained for this paper use approximately 1% of the parameters and take less than 1% of the GPU flop hours to train.

We see that the supervised models evaluated demonstrate shortcomings in performance on pitch ($NSynth_P$) and key (GS_{Key}) tasks. This corroborates the findings in [24] that models trained on tags do not perform well in this task, and the findings in [2] that supervised pre-trained models do not generalize as well to novel tasks. We note that there are no pitch or key labels in the Musicset dataset, and in fact, many of the labels employed (e.g., genre, mood, etc.) require the model to be somewhat agnostic to such information. It is yet to be seen whether including such information in the labels of a pre-training dataset would improve the generalizability of such models, though this is outside of the scope of the current work. Interestingly, we see that unsupervised models trained specifically on music data show significant improvements in key estimation.

When comparing the domain of the pre-training dataset features in the models of Musicset-USmall and Audioset-USmall we see that in all cases, in-domain data (music) results in a better performing model. We conjecture that this may be due to (a) increasing the chances of non-trivial negative examples within each batch relative to each positive pair, due to the homogeneity of the dataset, and (b) music is more spectrally dense and correlated than other common noise sources employed in mixup augmentation, such as speech or various environmental noises.

5. CONCLUSIONS

In this work we investigated both unsupervised and supervised learning strategies, to produce compact audio representations that may be deployed across industry-scale audio catalogs for a range of downstream use cases. We observed that supervised training on large scale expert-annotated music data achieves SOTA results in music tagging. We see that unsupervised learning on datasets of the same scale also achieves excellent performance, across a wider range of tasks than supervised learning, particularly on those that involve information not represented in the supervised dataset. Finally, we see that for unsupervised learning, restricting the domain of the pre-training dataset to music results in improvements in model performance.

6. ACKNOWLEDGEMENTS

The authors would like to thank the Pandora music analyst team for their years of dedicated and careful work that enables us to investigate supervised machine learning systems at modern levels of quality and scale.

7. REFERENCES

[1] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen, "Slow-fast auditory streams for audio recognition," in *ICASSP*, 2021, pp. 855–859.

[2] L. Wang, P. Luc, Y. Wu, A. Recasens, L. Smaira, A. Brock, A. Jaegle, J.-B. Alayrac, S. Dieleman, J. Carreira *et al.*, "Towards learning universal audio representations," in *ICASSP*, 2022, pp. 4593–4597.

[3] J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," *ISMIR*, 2018.

[4] L. Wang, P. Luc, A. Recasens, J.-B. Alayrac, and A. Oord, "Multimodal self-supervised learning of general audio representations," *arXiv preprint arXiv:2104.12807*, 2021.

[5] A. Jansen, M. Plakal, R. Pandya, D. P. Ellis, S. Hershey, J. Liu, R. C. Moore, and R. A. Saurous, "Unsupervised learning of semantic audio representations," in *ICASSP*, 2018, pp. 126–130.

[6] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[7] M. Tagliasacchi, B. Gfeller, F. de Chaumont Quitry, and D. Roblek, "Pre-training audio representations with self-supervision," *IEEE Signal Processing Letters*, vol. 27, pp. 600–604, 2020.

[8] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive learning of general-purpose audio representations," in *ICASSP*, 2021, pp. 3875–3879.

[9] L. Wang and A. Oord, "Multi-format contrastive learning of audio representations," *NeurIPS Workshops*, 2020.

[10] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," *ISMIR*, 2021.

[11] J. Lee, J. Park, K. L. Kim, and J. Nam, "SampleCNN: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, vol. 8, no. 1, p. 150, 2018.

[12] Q. Huang, A. Jansen, L. Zhang, D. P. Ellis, R. A. Saurous, and J. Anderson, "Large-scale weakly-supervised content embeddings for music recommendation and tagging," in *ICASSP*, 2020, pp. 8364–8368.

[13] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio spectrogram transformer," *Interspeech*, 2021.

[14] W.-T. Lu, J.-C. Wang, M. Won, K. Choi, and X. Song, "SpecTNT: a time-frequency transformer for music audio," *ISMIR*, 2021.

[15] M. Won, K. Choi, and X. Serra, "Semi-supervised music tagging transformer," *ISMIR*, 2021.

[16] J. Yan, Y. Song, W. Guo, L.-R. Dai, I. McLoughlin, and L. Chen, "A region based attention method for weakly supervised sound event detection and classification," in *ICASSP*, 2019, pp. 755–759.

[17] A. T. Liu, S.-w. Yang, P.-H. Chi, P.-c. Hsu, and H.-y. Lee, "Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders," in *ICASSP*, 2020, pp. 6419–6423.

[18] Y. Gong, Y.-A. Chung, and J. Glass, "PSLA: Improving audio tagging with pretraining, sampling, labeling, and aggregation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3292–3306, 2021.

[19] 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.

[20] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient training of audio transformers with patchout," *arXiv preprint arXiv:2110.05069*, 2021.

[21] I. Manco, E. Benetos, E. Quinton, and G. Fazekas, "Learning music audio representations via weak language supervision," in *ICASSP*, 2022, pp. 456–460.

[22] M. Won, S. Oramas, O. Nieto, F. Gouyon, and X. Serra, "Multimodal metric learning for tag-based music retrieval," in *ICASSP*, 2021, pp. 591–595.

[23] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[24] R. Castellon, C. Donahue, and P. Liang, "Codified audio language modeling learns useful representations for music information retrieval," *ISMIR*, 2021.

[25] J. Kim, J. Urbano, C. Liem, and A. Hanjalic, "One deep music representation to rule them all? a comparative analysis of different representation learning strategies," *Neural Computing and Applications*, vol. 32, no. 4, pp. 1067–1093, 2020.

[26] HEAR Benchmark, https://hearbenchmark.com/, [Accessed: 2022-08-01].

[27] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong

semi-supervised learners," *Advances in neural information processing systems*, vol. 33, pp. 22 243–22 255, 2020.

[28] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020, pp. 1597–1607.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

[30] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *ICLR*, 2018.

[31] M. Bulent Sariyildiz, Y. Kalantidis, K. Alahari, and D. Larlus, "Improving the generalization of supervised models," *arXiv preprint arXiv:2206.15369*, 2022.

[32] 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 *ICASSP*, 2017, pp. 776–780.

[33] M. C. McCallum, "Unsupervised learning of deep features for music segmentation," in *ICASSP*, 2019, pp. 346–350.

[34] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging." in *ISMIR*, 2009.

[35] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[36] F. Korzeniowski, O. Nieto, M. McCallum, M. Won, S. Oramas, and E. Schmidt, "Mood classification using listening data," in *ISMIR*, 2021.

[37] J. Pons and X. Serra, "musicnn: Pre-trained convolutional neural networks for music audio tagging," *arXiv preprint arXiv:1909.06654*, 2019.

[38] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, "The MTG-Jamendo dataset for automatic music tagging," in *ICML Workshops*, 2019. [Online]. Available: http://hdl.handle.net/10230/42015

[39] P. Knees, Faraldo, P. Herrera, R. Vogl, S. Böck, F. Hörschläger, and M. Le Goff, "Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections," in *ISMIR*, 2015.

[40] F. Korzeniowski and G. Widmer, "End-to-end musical key estimation using a convolutional neural network," in *EUSIPCO*, 2017, pp. 966–970.

[41] M. Soleymani, M. N. Caro, E. M. Schmidt, C.-Y. Sha, and Y.-H. Yang, "1000 songs for emotional analysis of music," in *CrowdMM*, 2013, pp. 1–6.

[42] S. Oramas, F. Barbieri, O. Nieto Caballero, and X. Serra, "Multimodal deep learning for music genre classification," *Transactions of the International Society for Music Information Retrieval*, vol. 1, no. 1, pp. 4–21.

[43] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "BYOL for audio: Self-supervised learning for general-purpose audio representation," in *IJCNN*, 2021, pp. 1–8.

[44] E. Koh and S. Dubnov, "Comparison and analysis of deep audio embeddings for music emotion recognition," *arXiv preprint arXiv:2104.06517*, 2021.

[45] F. Weninger, F. Eyben, and B. Schuller, "On-line continuous-time music mood regression with deep recurrent neural networks," in *ICASSP*, 2014, pp. 5412–5416.

[46] Pioneer DJ, https://rekordbox.com, [Accessed: 2016-09-12].

[47] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of CNN-based automatic music tagging models," *arXiv preprint arXiv:2006.00751*, 2020.

[48] G. Bernardes, M. E. Davies, and C. Guedes, "Automatic musical key estimation with adaptive mode bias," in *ICASSP*, 2017, pp. 316–320.

[49] D. Knox, T. Greer, B. Ma, E. Kuo, K. Somandepalli, and S. Narayanan, "Mediaeval 2020 emotion and theme recognition in music task: Loss function approaches for multi-label music tagging." in *MediaEval*, 2020.

GENERATING COHERENT DRUM ACCOMPANIMENT WITH FILLS AND IMPROVISATIONS

Rishabh Dahale [1] **Vaibhav Talwadker**[1] **Preeti Rao**[1] **Prateek Verma** [2]
[1] Department of Electrical Engineering, Indian Institute of Technology Bombay, India
[2] Stanford University
dahalerishabh1@iitb.ac.in, talwadkerv@gmail.com, prao@ee.iitb.ac.in, prateekv@stanford.edu

ABSTRACT

Creating a complex work of art like music necessitates profound creativity. With recent advancements in deep learning and powerful models such as transformers, there has been huge progress in automatic music generation. In an accompaniment generation context, creating a coherent drum pattern with apposite fills and improvisations at proper locations in a song is a challenging task even for an experienced drummer. Drum beats tend to follow a repetitive pattern through stanzas with fills/improvisation at section boundaries. In this work, we tackle the task of drum pattern generation conditioned on the accompanying music played by four melodic instruments – Piano, Guitar, Bass, and Strings. We use the transformer sequence to sequence model to generate a basic drum pattern conditioned on the melodic accompaniment to find that improvisation is largely absent, attributed possibly to its expectedly relatively low representation in the training data. We propose a novelty function to capture the extent of improvisation in a bar relative to its neighbors. We train a model to predict improvisation locations from the melodic accompaniment tracks. Finally, we use a novel BERT-inspired in-filling architecture, to learn the structure of both the drums and melody to in-fill elements of improvised music.

1. INTRODUCTION

Songs in popular music genres like rock are typically split into different sections such as the verse, bridge, and chorus. While the primary task of a drummer is to play in time, it is also important for the drummer to be consistent with the song structure. Traditionally fills, or short groups of notes, are played as the song transitions from one section to another (say, verse to chorus). Thus, it can serve as an indicator to the audience as well as the band, of an upcoming transition in the song. The duration of fills generally tends to be only a few beats long, no more than the length of a bar. Even though they are rare events in a drum track, fills are an important part of the overall aesthetics. Beyond signaling transitions, drum fills can also be played in sections where the accompanying instrumentation is sparse.

Motivated by the above, we improve the quality of the generated drum tracks from seq-to-seq models by incorporating fills/improvisations towards our overall goal of accompaniment generation. This is achieved via the following three distinct stages:

1. **Basic Drum Pattern Generation:** Using a seq-to-seq model to generate a drumbeat as the accompaniment to given melodic instrument tracks of guitar, bass, strings, and piano (i.e. the Melodic Accompaniment - MA).

2. **Improvisation Location Detection:** Detecting explicitly the position of improvisation from the MA using a self-similarity function and mini BERT model.

3. **Generating Improvised Bars:** Generating the fills in the previously detected bars.

Our main contributions are: (i) We show that traditional attention-based transformer architectures fail to capture the "improvisation" due to implicit data imbalance. (ii) We also show that the sampling-based approaches fail to produce a variation of pattern at the right location. To mitigate this, we learn to predict where to improvise directly from the melody tracks using powerful self-attention-based architectures. (iii) We propose a novel in-filling approach, inspired by BERT that can look at the context of drums and the context of melody and use it to generate the improvised bars. (iv) We demonstrate an MLP-based synthesis module for drum improvisation generation from a latent code. For simplicity we have ignored the dynamic (velocities) in the generated drum patterns of this work.

2. RELATED WORKS

Since the introduction of the Transformer architecture [1], there has been a growing interest in this model for sequential task modeling like in NLP and music generation. The architecture uses an attention mechanism to learn long-term patterns and can easily surpass dilated convolutional-based methods such as WaveNet [2]. They achieve state-of-the-art performance in a variety of natural language problems [3], music/audio understanding [4] and generation tasks [5, 6]. However, for generative models, the decoding strategy still remains an open question. Even though high-quality models can be obtained by the use of likelihood as the training objective, likelihood maximiza-

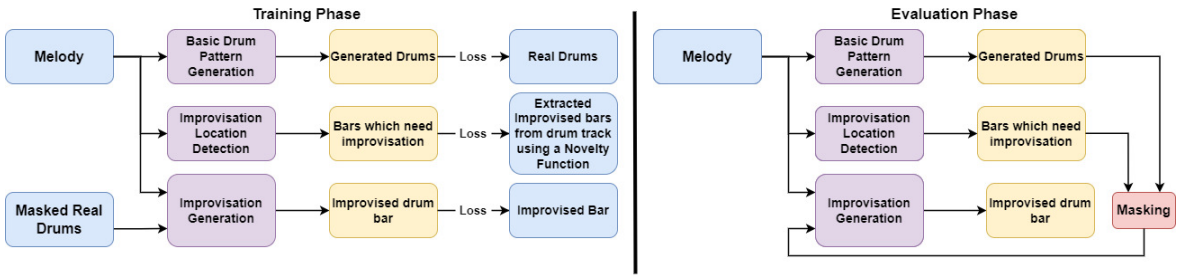

Figure 1. System Overview - Pipeline used for training (left half) of each individual module; combined pipeline (right half) for the evaluation phase.

tion methods like beam search lead to degeneration [7]. To solve this issue, many researchers use sampling-based methods like temperature sampling, top-k sampling, and nucleus sampling [7–9]. Despite all of the success of sequential modeling recently, there still exist many issues that are relevant to our current work such as understanding rare words or sparsely occurring events of interest [10]. Another major problem is the presence of biases in the generated output, as they mimic the distribution present in the training datasets [11]. These issues are ubiquitous across datasets and modalities and are implicit in our task due to the low representation of improvised bars. We here show how these constraints and biases affect the quality of drum generation, and the steps we take to mitigate them.

Conditioned drum beat generation is an important subtask of music generation. Wei et al. [12] observed that the polyphonic melody self-similarity matrix (SSM) is structurally similar to the drum SSM. They used this to predict the drum SSM from the melody SSM and used this intermediate representation to generate the drumbeat. While they used the audio form of the melody as the input, the symbolic domain also offers opportunities for similar research. An example is the work of [13] using symbolic representation of the drum track to predict locations and generate improvisations for the drum track. In this work, we address drum generation conditioned on a melodic accompaniment track, bringing considerably more complexity to the problem.

In the symbolic domain, systems use a discretized representation such as MIDI tag and pianoroll representations that capture the essential information at the semantic level. Pianoroll is a score-like matrix representing a piece of music. Note pitch and time are represented by the vertical and horizontal axes, respectively. The velocities of the notes are represented by the values. The time is generally quantized on a sub-beat level and each instrument track is represented by a separate pianoroll matrix. Dong et al. [14] used this representation along with CNN-based GAN model for music generation. Another popular symbolic domain representation is the MIDI tag representation which we refer to as the "serialized grid representation". In this method, the input is represented by a sequence of MIDI-like tags. Huang et al. [6] used this MIDI tag representation with modified transformer architecture to generate long-duration piano music. Several modifications have also been proposed to this representation. Huang et al. [15] proposed a revamped MIDI (REMI) which introduced `DURATION`, `BAR` and `POSITION` tags to improve the quaity of generated music. Ren et al. [16] further modified this representation to include multi-track representation by introducing `TRACK` tag and used the Transformer-XL model to generate multitrack songs. Nuttall et al. [17] modified the MIDI tags of nine percussion instruments to represent notes being played by a triplet of pitch, velocity, and start time, and used it with the Transformer-XL model to sequentially generate the drum pattern. Thorn et al. [18] demonstrated three experiments with the Transformer-XL model with varying input and output representation and control. In two of the experiments, they tried to control the drum machine while in the third experiment they tried to generate the drum pattern directly.

While the Transformer-XL model facilitates the generation of longer duration (musical) sequences, they still suffer from the same issues of biases in dealing with the implicit data-imbalance that exists in the training dataset [19, 20]. As the specific focus of this work is to find ways to capture the rare events in the generated output, i.e., fills and improvisations, we consider only the necessary context for an improvised bar in the form of 11 bar segments of the songs.

3. DATASET

In this work, we use the Lakh Pianoroll Dataset (LPD-5 cleansed) [14], derived from the Lakh Midi Dataset [21] which is a collection of 21,425 multitrack pianorolls files consisting of following tracks: Piano, Guitar, Bass, Strings and Percussion. All the songs in this dataset are of 4/4-time signature i.e., all the songs contain 4 beats in a bar and the dimensionality of each bar in this dataset is 128 (pitch) x 96 (time steps) i.e. each beat is divided into 24 parts. Our input, which we call melodic accompaniment (MA), consists of notes played by the 4 melodic instruments and output is a percussion instrument pattern conditioned on the input which we call the percussion accompaniment (PA).

Evaluation of generative music systems faces harder challenges than that of image generation systems [22]. We expect our system to replicate the rhythmic consistency and diversity of the dataset. Any drum beat generation system must have correct onset locations in a beat. If the onsets are not properly matched, it appears as if the drums are lagging/leading the melody. We show that our model is able learn this by capturing the onset location distribution.