Table 1. Summary of the notations

Symbol	Definition	Dimension
a	an audio track	
$\mathbf{M}_a$	Mel-spectrogram of a	$\mathbb{R}^{96\times646}$
$\mathbf{e}_a$	Embedding of a	$\mathbb{R}^{256}$
u	a user	
$\mathbf{e}_u$	Embedding of u	$\mathbb{R}^{128}$
$\mathbf{g}_u$	Demographics data of u	$\mathbb{R}^{3}$
$\mathbf{d}_u^{(t)}$	Device data of u at time t	$\mathbb{R}^{8}$
c	a situation (or context)	
s	a stream, a tuple $(a, u, \mathbf{d}_u^{(t)}, c)$	

is not computed exclusively with the tracks included in our dataset. The computed embeddings will be published with the dataset for reproducibility.

For the duSP (estimation of $\hat{c}_{d,u}$), we use the basic demographic data $\mathbf{g}_u$ of the user recorded during registration. This data is composed of: |age,country,gender|. While this data selection is prone to errors and short of fully representing the users, it is consistent with our requirements of using basic always-available data.

Device data $\mathbf{d}_u^{(t)}$. We collect only basic data sent by the device to the service and selected those that deemed relevant to the situation prediction. The data are: the time stamp (in local time), day of the week, device used and network used. Additionally, we extend the time/day data with circular representation of the time and day similar to the one used in [14]. The final feature vector representing device data is made of 8 features: linear-time, linear-day, circular-time-X, circular-time-Y, circular-day-X, circular-day-Y, device-type, network-type. The device-type can be: mobile, desktop (e.g. a laptop), or tablet. The network-type can be: mobile (a connection through cellular data), wifi (a WiFi connection), LAN (a connection through wired Ethernet), or plane (an offline stream from a device without a connection).

4. COLLECTING THE DATA

Pichl et al. and Ibrahim et al. proposed methods for labelling streaming sessions with a situational tag by leveraging playlist titles [2, 12]. Although sometimes prone to errors and false positives, playlist titles provide an appropriate proxy for labelling streams with tags [2, 12]. Users create playlists with a common theme according to their use [12]. One common theme of these playlists is the listening situation.

First, we collected a set of situational keywords used previously in the literature [1, 11, 15]. We extended these keywords by adding synonyms and hashtags that are frequently used on Twitter to refer to music listening. Afterwards, we retrieve from all public playlists from Deezer [3], an online music streaming service we were given access to, those playlists that include any of the keywords in their "stemmed" title. We then filtered out playlists that contained more than 100 tracks [4] or where a single artist or album represented more than 25% of the playlist, similar to [2].

From the extensive list of situational keywords and their corresponding playlists, we settled on three different subsets with an increasing number C of situational tags (4, 8, and 12): work, gym, party, sleep | morning, run, night, dance | car, train, relax, club.

These tags were selected by popularity [5]. We used these situations as independent tags without attempting to merge potentially similar activities and places (e.g. "party" and "dance"). Working with three situational tag sets (of increasing C) allowed us to observe how the system performs as the complexity of the problem increases.

We then focused on the users who actively listened to these playlists and retrieve the device data of those users while they were actively listening to the playlist. This resulted in a set of streams each described by an audio-track a, a user u, a playlist with a situational keyword c in the title, along with the device data $\mathbf{d}_u^{(t)}$ sent during this stream. Note that an audio-track/user/device triplet have a joint tag, none of them are tagged individually.

To ensure high quality data, we selected the month of August 2019 for inspection, because this period had more stable use patterns, before the Covid-19 pandemic. We had access to data from two locations: France and Brazil. These two locations were provided because they have the most active users in Deezer, while being in two distinctive time zones and seasons. This allowed us to perform our study on diverse data with different sources and patterns. We release the dataset [6] along with the code [7].

4.1 Dataset Analysis

As a sanity check on the collected data, we plot the distribution of the situations c across the different device-data. Figure 3 shows the ratio of the used network-type to connect to the service across situations c. We observe variations that correspond to what is expected from each situation, i.e. outdoors vs. indoors. However, we also find certain networks that do not match the expectations, e.g. LAN connections in a car situation, which represents noise in the dataset that can be a continuation of already existing sessions that moved indoors. Figure 4 represents the used device-type across situations c. We notice that most users overwhelmingly use mobile device in most cases, with small variations that also match expectations of indoor and outdoor situations. Finally, Figure 5 shows the distribution of all situations for each hour of the day. Similarly, we find predictable patterns for each situation ranging from night-related situation in the early hours that gradually progress as the time passes. These patterns support

[3] www.deezer.com

[4] to increase the chance that playlists reflect a selection of situation-related tracks, and not randomly added ones

[5] Estimated as the number of corresponding playlists in the service catalogue.

[6] https://zenodo.org/record/5552288

[7] https://github.com/Karimmibrahim/Situational_Session_Generator

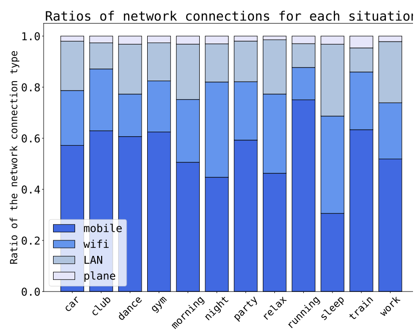

Figure 3. Network across situations c

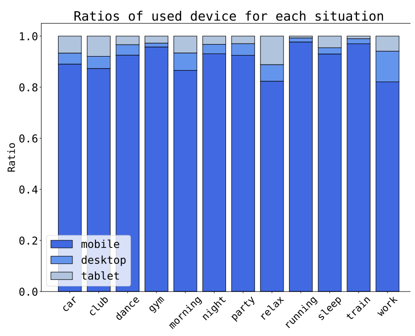

Figure 4. Device across situations c

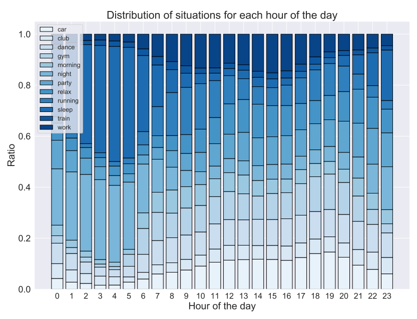

Figure 5. Distributions of situations c over hours of the day

the hypothesis of using playlist titles as proxy for inferring the actual listening situation.

5. DETAILED MODELS DESCRIPTION

5.1 Audio+User based Situation Prediction (`auSP`)

The `auSP` estimates the probability of each situation $c \in \{1 \dots C\}$ given a pair (track a represented by $\mathbf{M}_a$, user u represented by e_u): $P(c|\mathbf{M}_a, \mathbf{e_u})$. It is implemented as a Deep Neural Network $\hat{c}_{a,u} = f_\theta(\mathbf{M}_a, \mathbf{e}_u)$ with softmax output and trainable parameters θ. To train it we use the set of training streams represented as tuples $(\mathbf{M}_a, \mathbf{e}_u, c)$ We train it by minimizing the categorical cross-entropy $\mathcal{L}(\hat{\mathbf{c}}_{a,u}, \mathbf{c}, \theta)$ where $\hat{\mathbf{c}}_{a,u}$ is the estimated probability and $\mathbf{c}$ the one-hot-encoded ground-truth.

Practical implementation. The **audio input**, $\mathbf{M}_a$, is passed to a batch normalization layer then to 4 layers each made of a convolutional (CNN) and a Max-pooling operation. The CNNs have various numbers of filters (32, 64, 128, 256) but each with the same size (3x3). They are each followed by a ReLU. All Max-poolings are (2x2). The flattened output of the last CNN layer is passed to a fully connected (FC) layer with 256 units followed by a ReLU. The output of the audio branch $\mathbf{e}_a$ is a 256-d audio embedding vector $\mathbf{e_a}$. The **user input**, $\mathbf{e}_u$, is processed through 2 FC layers each with a ReLU. This output is then concatenated with $\mathbf{e}_a$ and passed to a FC layer with ReLU activation, and a dropout (with 0.3 ratio) for regularization. The final layer is made of C output units with a Softmax activation function, where C is the number of situations to be predicted. We train the model until convergence by minimizing the categorical cross entropy, optimized with Adam [16] and a learning rate initialized to 0.1 with an exponential decay every 1000 iterations.

5.2 Device+User based Situation Prediction (`duSP`)

The `duSP` estimates the probability of each situation $c \in \{1 \dots C\}$ given a pair (device-data d represented by $\mathbf{d}_u^{(t)}$, user u represented by $\mathbf{g}_u$): $P(c|\mathbf{d}_u^{(t)}, \mathbf{g}_u)$. It is implemented as a function $f_\gamma(\mathbf{d}_u^{(t)}, \mathbf{g_u})$ with Softmax output and trainable parameters γ. To train it we use the set of training streams represented as tuples $(\mathbf{g}_u, \mathbf{d}_u^{(t)}, c)$

Practical implementation. In choosing a real-time "light" `duSP` model, we prioritize the computational complexity requirements over accuracy. The low dimensional input features (11-d = 8 device features + 3 demographic features) already provide a strong case for the investigated models. For our implementation, we experimented with different classifiers: Decision Trees, K-Nearest Neighbors, and eXtreme Gradient Boosting (XGBoost) [17]. While all gave comparable results, we chose XGBoost for its consistent performance across splits and different evaluation scenarios. Similar to the autotagger model, the output predictions depend on the number C of situations in the dataset.

6. EVALUATION

We evaluate here the performance of our system which aims at proposing for a given user u a session (sequence of audio-tracks a) that fits their current situation c. For this, we first evaluate the performance of the two branches of our system (`auSP` and `duSP`) to correctly estimate the situation c. We evaluate this using various numbers of situations: $C \in \{4, 8, 12\}$. To evaluate the `auSP`, we use the common AUC (Area Under Roc Curve) and Accuracy performance measures. To evaluate the `duSP`, we use the Accuracy but also the Accuracy@K. This measures the capability of `duSP` to include the correct situation in the top K predictions. We then evaluate the global system by measuring the overlap of correct predictions between the `auSP` and `duSP` branches. This accuracy can be interpreted as the ratio of existing streams that would have occurred in these sessions if the playlists were generated with this system instead.

6.1 Scenario

We approach the evaluation of this system from two different perspectives: 1) evaluating the system on its capability of learning and generalizing, 2) evaluating the proposed system in a stable use-case with frequent users/tracks.

We simulate these scenarios through a different split criteria for the test-set. Let the full set of streams in our collected dataset S, where each stream s has a user u and a track a. We will be referring to the training-set as S_{train}, the test-set as S_{test}, the set of unique users in training and testing as U_{train} and U_{test} respectively, and similarly the unique audio-tracks in the splits as A_{train} and A_{test}.

To evaluate the system intelligence and fit to the data, we restrict the evaluation splits to include either: 1) new users (cold-user case): $S_{test} = \{s|u \notin U_{train}, a \in A_{train}\}$, 2) new tracks (cold-track case): $S_{test} = \{s|u \in U_{train}, a \notin A_{train}\}$. We exclude the specific case of both new tracks and new users because splitting the data with only new user/track pairs in the testset is difficult and rare to find. Additionally, recommending a new track to a new user is not a common nor practical scenario to use for evaluating a system.

To evaluate how the system would perform in a regular use-case (warm case): $S_{test} = \{s|u \in U_{train}, a \in A_{train}, s \notin S_{train}\}$. The regular use-case does not restrict the system to neither new users nor tracks. However, the test-set contains exclusively new streams, i.e. (user/track) pairs, not present in the training-set. The evaluation of this regular use-case is relatively complex and includes several entwined evaluation criteria. The goal is to compare the overlap of generated sessions with groundtruth sessions.

6.1.1 auSP Evaluation

The results for the auSP can be found in Table 2.

As shown, the model can reach satisfying performance relative to the evaluation scenario. In terms of AUC, the model's fit for both new users and tracks in the cold user/track splits is not significantly impaired compared to the warm case. The performance decreases evidently as the problem gets harder with more situations C to tag, though in some cases it increases given the increase of dataset size from additional situations. In terms of accuracy, the model's performance in the intended use-case, i.e. warm case, is satisfying (Accuracy above 70). That is to say, the system can correctly tag around two thirds of the user/track listen streams with their correct situational use, when it has seen the user or the track before, but not jointly.

Note that this accuracy was computed by selecting the most probable situation from the predictions. While the high values of AUC (above 0.94) suggest a threshold optimization is needed for each class, in real use-case we do not necessarily need a threshold. The prediction probability could be used directly to retrieve tracks, e.g. by ranking tracks with the prediction probabilities and include top ranked tracks in the generated sessions. However, this max-probability threshold is needed for further evaluations with the situation predictor and with the sequential retrieval model.

Additionally, Figure 6 represents the confusion matrix obtained in the $C = 12$ and warm case.

6.1.2 duSP Evaluation

The results for the duSP can be found in Table 3. We find that predicting the situation for new users becomes noticeably harder. In the case of C=12 situations, the system was able to correctly predict the situation for only 25% of the streams. However, when the system is allowed to make multiple guesses (Accuracy@3), the accuracy evidently in-

Table 2. Results of the auSP evaluated with AUC and Accuracy in the three evaluation protocol splits (cold-user, cold-track, and warm case) and the three subsets of situations (4, 8, and 12). The results are shown as mean(std.).

C	AUC		
	Cold User	Cold Track	Warm
4	0.889 (.009)	0.873 (.013)	0.959 (.013)
8	0.815 (.005)	0.866 (.007)	0.945 (.007)
12	0.852 (.004)	0.824 (.012)	0.941 (.012)
C	Accuracy		
	Cold User	Cold Track	Warm
4	69.72 (1.07)	63.77 (2.33)	83.75 (2.33)
8	47.56 (0.53)	52.44 (2.31)	70.81 (1.45)
12	52.68 (1.25)	37.61 (3.47)	69.14 (3.79)

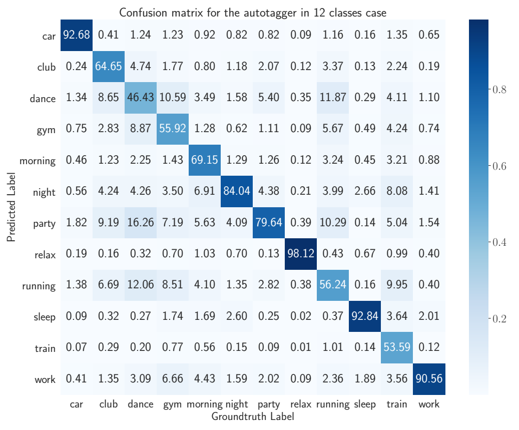

Figure 6. Confusion Matrix of the auSP in the case C=12 and warm case

creases. In the case where the user is to make the last decision, the system is able to include the correct situation in the top 3 suggestions 96%, 80%, and 68% in the cases of C =4, 8, and 12 situations respectively. The choice of K, when evaluating with accuracy@K, can be obviously changed, and the performance will increase as K increases. We choose to display the results for K=3 since 3 is around the number of visible items in the carousels displayed by most streaming services on the suggestions screen on mobile devices.

Additionally, Figure 7 shows the confusion matrix obtained in the C=12 situations and warm case. We observe that the confusion is mostly coherent with the statistic shown earlier of the distribution of situations with the device data. Situations that are likely to originate with similar device data are harder to discriminate than the rest. For example, we observe a cluster of night-related situations including night, sleep, and relax situations. Similarly, outdoors situation are also often confused together. Discriminating those situations is hindered by the limited data available. However, the convenience of recommending top k situations provides as easy solution to further discriminate between these similar situations.

Table 3. Results of the `duSP` evaluated with Accuracy and Accuracy@3 in the three evaluation protocol splits (cold-user, cold-track, and warm case) and the three subsets of situations (4, 8, and 12). The results are shown as mean(std.).

C	Accuracy		Accuracy @3	
	Cold User	Warm	Cold User	Warm
4	47.46 (0.98)	66.96 (0.39)	90.51 (0.31)	96.3 (0.1)
8	30.95 (0.89)	49.23 (0.16)	64.11 (1.42)	79.62 (0.13)
12	25.00 (0.29)	39.92 (0.13)	52.04 (0.61)	67.62 (0.21)

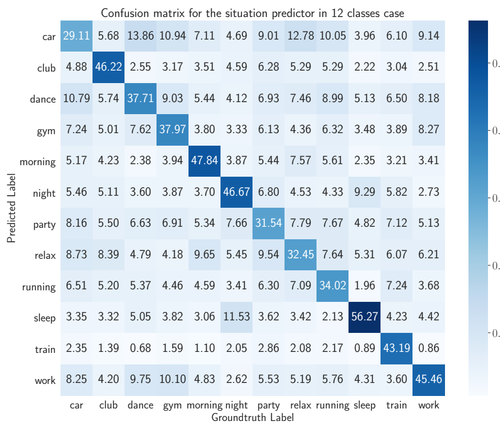

Figure 7. Confusion Matrix of the `duSP` with C=12 and warm case

Finally, to evaluate the challenge in classifying situations from multiple sources, we compare between the evaluation results in each location (France, Brazil) separately. We compare between two different cases: 1) a model trained globally on the data from both locations but tested locally, 2) a model trained locally on each location independently and tested on the corresponding location. Table 4 shows the results for this evaluation setting. We find that training the models locally slightly improves the results, but not significantly. This suggests that using a single unique model for all locations gives comparable results to using multiple local models. We also observe a clear distinction in the accuracy between the two locations, where Brazil scores higher than France in all cases. This is due to the larger number of users in our dataset who are in France, i.e. there are more users with more distinct patterns in the France case.

Table 4. Evaluation results of the globally and locally trained models for each of the two locations in our dataset, France and Brazil, evaluated with accuracy at each subset of situations in the warm case. The results are shown as mean(std.).

C	France		Brazil	
	Global	Local	Global	Local
4	53.4 (0.2)	55.1 (0.2)	59.2 (0.2)	61.7 (0.2)
8	35.8 (0.1)	36.8 (0.1)	45.9 (0.1)	48.2 (0.1)
12	27.6 (0.1)	28.2 (0.1)	39.6 (0.2)	41.8 (0.1)

Table 5. The joint evaluation results of the `auSP` and `duSP` and their overlapping predictions evaluated with Accuracy in the three evaluation protocol splits (cold-user, cold-track, and warm case) and the three subsets of situations (4, 8, and 12). The results are shown as mean(std.).

Model	Cold Users	Cold Tracks	Warm Case
		4 Situations	
`auSP`	69.73 (1.07)	63.78 (2.33)	83.75 (2.33)
`duSP`	47.46 (0.98)	66.81 (0.35)	67.20 (0.26)
Overlap	36.22 (1.27)	44.60 (1.01)	**58.92** (1.71)
		8 Situations	
`auSP`	47.56 (0.53)	52.44 (2.31)	70.81 (1.45)
`duSP`	30.95 (0.89)	49.13 (0.24)	49.35 (0.19)
Overlap	17.77 (0.49)	28.94 (1.24)	39.52 (1.27)
		12 Situations	
`auSP`	52.68 (1.25)	37.61 (3.47)	69.14 (3.79)
`duSP`	25.00 (0.29)	39.05 (0.31)	39.19 (0.14)
Overlap	16.19 (0.32)	18.75 (1.63)	31.26 (1.30)

6.1.3 Joint Evaluation

The results for the joint system can be found in Table 5. As we can see, each variable in our evaluation influences the performance of the system. The most influential parameter is the number C of potential situations. As the complexity increases, we find the accuracy of the model decreasing: from 58% in the case C=4 with no new users or tracks to 16% in the case C=12 with cold scenarios. Additionally, we find the expected variation in performance between the cold cases and the warm case of intended use. We observe how the drop in the performance of the `auSP` and `duSP`, on new users/tracks, negatively affects the joint system performance.

However, in the harder evaluation case of generating a situational playlists with only 1 guess allowed out of C=12, the proposed system would have been able to include at least a third of the actual listened tracks (31.26%) in those playlists, while pushing them to the user at the exact listened time.

7. CONCLUSION

In this study, we address the problem of the unobserved listening situation which influences the users' preferences. We proposed a two-branch framework to predict when a situation is being experienced based on the device data, while simultaneously autotagging the music tracks with their intended listening situation in a personalized manner. Through the proposed approach, users could access a set of predicted potential situations. These situations are also associated with a set of tracks "likely" to be listened to by the user. This likelihood is estimated using an autotagger trained on predicting the situational use of tracks, given a specific user and his/her listening history. We evaluated each of our system's blocks individually and combined. The evaluation results indicated that the system is capable of learning personalized patterns for users, which can be employed to provide contextual music recommendation.

8. REFERENCES

[1] A. C. North and D. J. Hargreaves, "Situational influences on reported musical preference." *Psychomusicology: A Journal of Research in Music Cognition*, vol. 15, no. 1-2, p. 30, 1996.

[2] K. Ibrahim, E. Epure, G. Peeters, and G. Richard, "Should we consider the users in contextual music auto-tagging models?" in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[3] A. E. Greasley and A. Lamont, "Exploring engagement with music in everyday life using experience sampling methodology," *Musicae Scientiae*, vol. 15, no. 1, pp. 45–71, 2011.

[4] M. Gorgoglione, U. Panniello, and A. Tuzhilin, "The effect of context-aware recommendations on customer purchasing behavior and trust," in *Proceedings of the fifth ACM conference on Recommender systems*, 2011, pp. 85–92.

[5] C. Hansen, C. Hansen, L. Maystre, R. Mehrotra, B. Brost, F. Tomasi, and M. Lalmas, "Contextual and sequential user embeddings for large-scale music recommendation," in *Fourteenth ACM Conference on Recommender Systems*, 2020, pp. 53–62.

[6] M. Kaminskas and F. Ricci, "Contextual music information retrieval and recommendation: State of the art and challenges," *Computer Science Review*, vol. 6, no. 2-3, pp. 89–119, 2012.

[7] Ò. Celma and X. Serra, "Foafing the music: Bridging the semantic gap in music recommendation," *Journal of Web Semantics*, vol. 6, no. 4, pp. 250–256, 2008.

[8] K. Ibrahim, J. Royo-Letelier, E. Epure, G. Peeters, and G. Richard, "Audio-based auto-tagging with contextual tags for music," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020.

[9] M. Schedl, A. Flexer, and J. Urbano, "The neglected user in music information retrieval research," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 523–539, 2013.

[10] F. Korzeniowski, O. Nieto, M. McCallum, M. Won, S. Oramas, and E. Schmidt, "Mood classification using listening data," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR*, 2020.

[11] X. Wang, D. Rosenblum, and Y. Wang, "Context-aware mobile music recommendation for daily activities," in *Proceedings of the 20th ACM international conference on Multimedia*, 2012, pp. 99–108.

[12] M. Pichl, E. Zangerle, and G. Specht, "Towards a context-aware music recommendation approach: What is hidden in the playlist name?" in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, 2015, pp. 1360–1365.

[13] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[14] P. Herrera, Z. Resa, and M. Sordo, "Rocking around the clock eight days a week: an exploration of temporal patterns of music listening," in *Proceedings of the 1st Workshop On Music Recommendation And Discovery (WOMRAD), ACM RecSys, 2010, Barcelona, Spain*, 2010.

[15] M. Gillhofer and M. Schedl, "Iron maiden while jogging, debussy for dinner?" in *MultiMedia Modeling*, X. He, S. Luo, D. Tao, C. Xu, J. Yang, and M. A. Hasan, Eds. Cham: Springer International Publishing, 2015, pp. 380–391.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[17] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

JUKEDRUMMER: CONDITIONAL BEAT-AWARE AUDIO-DOMAIN DRUM ACCOMPANIMENT GENERATION VIA TRANSFORMER VQ-VAE

Yueh-Kao Wu
Academia Sinica
yk.lego09@gmail.com

Ching-Yu Chiu
National Cheng Kung University
x2009971@gmail.com

Yi-Hsuan Yang
Taiwan AI Labs
yhyang@ailabs.tw

ABSTRACT

This paper proposes a model that generates a drum track in the audio domain to play along to a user-provided drum-free recording. Specifically, using paired data of drumless tracks and the corresponding human-made drum tracks, we train a Transformer model to improvise the drum part of an unseen drumless recording. We combine two approaches to encode the input audio. First, we train a vector-quantized variational autoencoder (VQ-VAE) to represent the input audio with discrete codes, which can then be readily used in a Transformer. Second, using an audio-domain beat tracking model, we compute beat-related features of the input audio and use them as embeddings in the Transformer. Instead of generating the drum track directly as waveforms, we use a separate VQ-VAE to encode the mel-spectrogram of a drum track into another set of discrete codes, and train the Transformer to predict the sequence of drum-related discrete codes. The output codes are then converted to a mel-spectrogram with a decoder, and then to the waveform with a vocoder. We report both objective and subjective evaluations of variants of the proposed model, demonstrating that the model with beat information generates drum accompaniment that is rhythmically and stylistically consistent with the input audio.

1. INTRODUCTION

Deep generative models for musical audio generation have witnessed great progress in recent years [1–8]. While models for generating symbolic music such as MIDI [9–12] or musical scores [13] focus primarily on the *composition* of musical content, an audio-domain music generation model deals with *sounds* and thereby has extra complexities related to timbre and audio quality. For example, while a model for generating symbolic guitar tabs can simply consider a guitar tab as a sequence of notes [11], a model that generates audio recordings of guitar needs to determine not only the underlying sequence of notes but also the way to render (synthesize) the notes into sounds. Due to the complexities involved, research on deep generative models for musical audio begins with the simpler task of synthesizing individual musical notes [1–3], dispensing the need to consider the composition of notes. Follow-up research [4–6] extends the capability to generating musical passages of a single instrument. The Jukebox model [7] proposed by OpenAI greatly advances the state-of-the-art by being able to, quoting their sentence, "generate high-fidelity and diverse songs with coherence up to multiple minutes." Being trained on a massive collection of audio recordings with the corresponding lyrics but not the symbolic transcriptions of music, Jukebox generates multi-instrument music as raw waveforms directly without an explicit model of the underlying sequence of notes.

This work aims to improve upon Jukebox in two aspects. First, the backbone of Jukebox is a hundred-layer Transformer [14, 15] with billions of parameters that are trained with 1.2 million songs on hundreds of NVIDIA V100 GPUs for weeks at OpenAI, which is hard to reproduce elsewhere. Inspired by a recent Jukebox-like model for singing voice generation called KaraSinger [8], we instead build a light-weight model with only 25 million parameters by working on Mel-spectrograms instead of raw waveforms. Our model is trained with only 457 recordings on a single GeForce GTX 1080 Ti GPU for 2 days.

Second, and more importantly, instead of a fully autonomous model that makes a song from scratch with various instruments, we aim to build a model that can work cooperatively with human, allowing the human partner to come up with the musical audio of *some* instruments as input to the model, and generating in return the musical audio of *some other* instruments to accompany and to complement the user input, completing the song together. Such a model can potentially contribute to human-AI co-creation in songwriting [16] and enable new applications.

In technical terms, our work enhances the controllability of the model by allowing its generation to be steered on a user-provided audio track. It can be viewed as an interesting sequence-to-sequence problem where the model creates a "target sequence" of music that is to be played along to the input "source sequence." Besides requirement on audio quality, the coordination between the source and target sequences in terms of musical aspects such as style, rhythm, and harmony is also of central importance.

We note that, for controllability and the intelligibility of the generated singing, both Jukebox [7] and KaraSinger [8] have a lyrics encoder that allows their generation to be steered on textual lyrics. While being technically similar, our *accompaniment generation* task ("audio-to-audio") is different from the lyric-conditioned generation task ("text-to-audio") in that the latter does not need to deal with the coordination between two audio recordings.

Specifically, we consider a *drum accompaniment generation* problem in our implementation, using a "drumless" recording as the input and generating as the output a drum track that involves the use of an entire drum kit. We use this as an example task to investigate the audio-domain accompaniment generation problem out of the following reasons. First, datasets used in musical source separation [17] usually consist of an isolated drum stem along with stems corresponding to other instruments. We can therefore easily merge the other stems to create paired data of drumless tracks and drum tracks as training data of our model. (In musical terms, drumless, or "Minus Drums" songs are recordings where the drum part has been taken out, which corresponds nicely to our scenario.) Second, we suppose a drum accompaniment generation model can easily find applications in songwriting [18], as it allows a user (who may not be familiar with drum playing or beat making) to focus on the other non-drum tracks. Third, audio-domain drum accompaniment generation poses interesting challenges as the model needs to determine not only the *drum patterns* but also the *drum sounds* that are supposed to be, respectively, rhythmically and stylistically consistent with the input. Moreover, the generated drum track is expected to follow a steady tempo, which is a basic requirement for a human drummer. We call our model the "JukeDrummer."

As depicted in Figure 1, the proposed model architecture contains an "audio encoder" (instead of the original text encoder [7, 8]) named the *drumless VQ encoder* that takes a drum-free audio as input. Besides, we experiment with different ways to capitalize an audio-domain beat and downbeat tracking model proposed recently [19] in a novel *beat-aware module* that extracts *beat-related information* from the input audio, so that the language model for generation (i.e., the Transformer) is better informed of the rhythmic properties of the input. The specific model [19] was trained on drumless recordings as well, befitting our task. We extract features from different levels, including low-level tracker embeddings, mid-level activation peaks, and high-level beat/downbeat positions, and investigate which one benefits the generation model the most.

Our contribution is four-fold. First, to our best knowledge, this work represents the first attempt to drum accompaniment generation of a full drum kit given drum-free mixed audio. Second, we develop a light-weight audio-to-audio Jukebox variant that takes an input audio of up to 24 seconds as conditioning and generates accompanying music in the domain of Mel-spectrograms (Section 3). Third, we experiment with different beat-related conditions in the context of audio generation (Section 4). Finally, we report objective and subjective evaluations demonstrating the effectiveness of the proposed model (Sections 6 & 7). [1] .

[1] We share our code and checkpoint at: `https://github.com/legoodmanner/jukedrummer`. Moreover, we provide audio examples at the following demo page: `https://legoodmanner.github.io/jukedrummer-demo/`

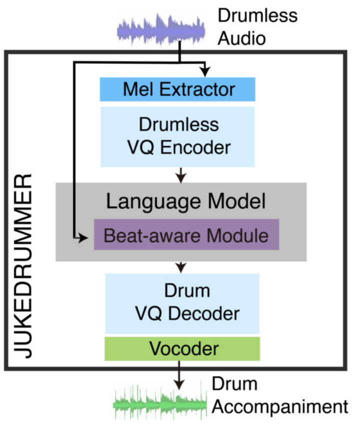

Figure 1: Diagram of the proposed JukeDrummer model for the inference stage. The training stage involves learning additional Drum VQ Encoder and Drumless VQ Decoder (see Figure 2) that are not used at inference time.

2. BACKGROUND

2.1 Related Work on Drum Generation

Conditional drum accompaniment generation has been studied in the literature, but only in the symbolic domain [20, 21], to the best of our knowledge. Dahale *et al.* [20] used a Transformer encoder to generate an accompanying symbolic drum pattern of 12 bars given a four-track, melodic MIDI passage. Makris *et al.* [21] adopted instead a sequence-to-sequence architecture with a bi-directional long short-term memory (BLSTM) encoder extracting information from the melodic input and a Transformer decoder generating the drum track for up to 16 bars in MIDI format. While symbolic-domain music generation has its own challenges, it differs greatly from the audio-domain counterpart studied in this paper, for it is not about generating sounds that can be readily listened to by human.

Related tasks that have been attempted in the literature with deep learning include symbolic-domain generation of a monophonic drum track (i.e., kick drum only) of multiple bars [4], symbolic-domain drum pattern generation [22–25], symbolic-domain drum track generation as part of a multi-track MIDI [26–29], audio-domain one-shot drum hit generation [30–34], audio-domain generation of drum sounds of an entire drum kit of a single bar [35], and audio-domain drum loop generation [36]. Jukebox [7] generates a mixture of sounds that include drums, but not an isolated drum track. By design, Jukebox does not take any input audio as a condition and generate accompaniments.

2.2 The Original Jukebox model

The main architecture of Jukebox [7] is composed of two components: a multi-scale vector-quantized variational autoencoder (VQ-VAE) [37–41] and an autoregressive Transformer decoder [14, 15]. The **VQ-VAE** is for converting a continuous-valued raw audio waveform into a sequence of so-called discrete *VQ codes*, while the **Transformer** establishes a language model (LM) of the VQ codes capable of generating new code sequences.