Automatic Live Music Soundchecking with Reference Audio on a Laptop

Matthew Keating
Dartmouth College
Hanover, New Hampshire, USA
matthew.b.keating.gr@dartmouth.edu

Michael Casey
Dartmouth College
Hanover, New Hampshire, USA
michael.a.casey@dartmouth.edu

Abstract

We introduce a live mixing system that gives visual recommendations directly to a guitarist while they play with a band, based on a reference track. Our work runs on a laptop computer and receives audio input through the laptop's microphone. We call this system SoLAR: **So**undchecking for **L**ive Band **A**udio with **R**eference. This work focuses on guitarist loudness control, but the framework is built to scale to other instruments and control parameters. SoLAR uses modern source separation techniques to get instrument stems from the reference audio. We introduce a perceptual algorithm for calculating which part of the Bark scale each instrument dominates to find a mix summarization. We fit a neural network model to predict the reference mix summarization. SoLAR makes recommendations for loudness adjustment by comparing the predicted reference mix to the current live mix in a policy function. We demonstrate that these recommendations improve on random guessing through simulations in a gymnasium environment. We describe the SoLAR graphical user interface and our design considerations. Then, we discuss a small formative study with guitarists. This study allowed us to iterate on the interface and showed promising early results for live mixing.[1]

Keywords

Mixing, Guitar, Live, Perceptual Audio, Machine Learning

1 Introduction

We focus this work on bands that play concerts at small venues and do not have a mixing board or a mixing engineer. When these musicians arrive at the venue, they must set up their instruments and amplifiers in a soundcheck. During a soundcheck, the band will select a song from their repertoire and play it, while one of the band members or a friend listens from the middle of the audience area.[2] The listener needs to be in the audience area because each band member is too close to their instrument to judge the total sound. The listener will provide verbal instructions to each band member, such as "turn the guitar down" or "I need to hear more vocals." Many of these instructions can be satisfied by turning a knob or adjusting one's playing style. The difficulty with this soundchecking method is that the listener needs to have music experience, and they need to understand the ideal mix the band is trying to achieve.

These adjustments are crucial for a strong performance. If a band makes a mistake soundchecking, the audience may not be able to hear the singer, the guitarist may hurt some ears, or the guitar solos could be imperceptible. These deficiencies would leave both the audience and band dissatisfied with the performance. Small shows are crucial for local communities and give musicians an opportunity to play for friends and neighbors. A musician can practice their instrument to mastery, but that skill only comes through at a show if their sound is audible.

Live soundchecking adjustments are also important because a mix is expressive and highly personalized. Any band has its *sound*, and that sound comes from the members' technique, the songs they play, and their mix. For instance, a grunge rock band may prefer to turn the guitar up, while a singer-songwriter could prefer to have their guitarist in the background. As such, there is no universal mix, and each decision in crafting a live mix changes the way a band sounds and allows for high expressivity.

This work focuses on loudness because gain adjustment has established simulation strategies. We study this system with the **guitarist's** actions because a guitarist has significant control over their loudness, through their amplifier knobs, the volume knob on their guitar, their playing style, and any pedals they use to modify their output signal, and the guitar is the first author's primary instrument.

SoLAR is a system that runs on a laptop computer and assists with mixing by making recommendations for loudness control given, as input, a reference recording that demonstrates an *ideal* mix. SoLAR learns the role each instrument should occupy in a perceptual mix from the reference audio. Then, as the guitarist plays, SoLAR makes recommendations for actions such as "turn up" or "stay there". The guitarist can improvise their part during soundchecking, and does not need to adhere to the original guitar part in the reference track. SoLAR increases the usability of an amplifier because SoLAR can give feedback to the performer on what the audience hears.

The novelty of our work is threefold:

(1) We provide live mixing recommendations directly to instrumentalists, who may not be able to afford a mixing engineer, where the only cost is a laptop computer
(2) We do not rely on a mixing board, and make recommendations that can be fulfilled with a simple amplifier knob turn or playing adjustment
(3) We use a reference audio track to allow for expressive control

2 Related Work

Automatic music mixing has been studied in a few main contexts. The main research formulation, typically called automatic mixing, works with mixing recorded music in a digital audio workstation (DAW). Mixing tools for live performance are typically further qualified with names like automatic live mixing or live audio

[1] https://github.com/mbkeating/SoLAR
[2] For example, this Reddit thread has a discussion on our soundchecking procedure https://www.reddit.com/r/livesound/comments/1d1sl7l/my_bands_first_show_is_this_week_in_a_small_bar/

NIME '26, June 23–26, 2026, London, UK

mixing. Standard automatic mixing is difficult because fully end-to-end approaches have control limitations, many audio effects are non-differentiable, and audio effects can interfere with source separation techniques [17]. Live audio mixing presents additional difficulties due to room acoustics and crowd noise. We expand on live audio mixing literature by working with reference audio and offering recommendations directly to instrumentalists.

Surveys on automatic mixing have found that mixing is considered an art form [22], therefore a method of musical expression. Similar studies have highlighted the usage of a reference song by a mixing engineer in mixing, a song that has a similar sound to the desired mix [23]. This reference song concept is central to the SoLAR workflow.

Automatic live mixing software typically seeks to set gain levels based on instrument identification and perceptual loudness measures [10, 11]. Recent work has focused on giving humans greater control over the mixing process. This work includes MixAssist, a dataset for language models to have step-by-step mixing controls, rather than full automation [4]. One particularly user-centric work is Channel-AI, which uses instrument recognition and instrument presets to build a smart mixing workstation for mixing engineers [21]. We take inspiration from the Wekinator as a user-centric AI system [6]. Other relevant NIME work includes live mixing work using Max/MSP virtual instruments and patches [1], guitar intent inference from muscle contractions [8], and brain-computer interfaces for accessible participation in mixing [24].

Audio source separation entails separating an input audio mixture into individual stems. We use Demucs in our system, which separates into drums, vocals, bass, and other, where other is guitar [5, 14]. Demucs also has a six-stem model that additionally separates guitar and piano. Our system can easily swap to the six stem model, but we had more reliable testing results with four stems. Recent work has iterated on general source separation [12, 15]; we used Demucs because the API is well documented, and Demucs is made specifically for instruments.

3 System Overview

Please see **figure 1** for the SoLAR system overview.

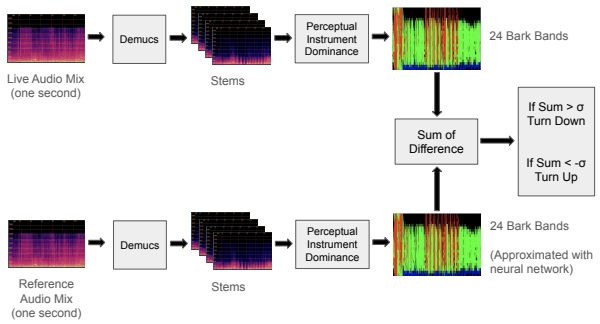

Figure 1: We start with live audio and a reference track. Both tracks get source separated and passed through our perceptual dominance algorithm. We use an element-wise difference to make recommendations for loudness adjustment. We set $\sigma = 3$ in section 6

4 Perceptual Audio

We use perceptual audio standards to summarize the mix from a reference track. This summarization process is intended to remove variation due to non-mix related decisions, such as the song melody or rhythm. In general, we know that our ears do not hear all frequencies equally, and instead experience masking in critical bands. Zwicker et al. experimentally derived the boundary frequencies for the first 24 critical bands [25]. If two different sine waves are played, and both have a frequency within the same critical band, the weaker frequency will be masked, meaning it will be less perceptible. Masking is a key concept behind the MP3 audio compression algorithm, which filters out audio data that we know will be masked [13].

We can summarize the mix by source separating the audio and finding which instrument dominates each critical band, causing the other instruments to be masked. Therefore, we are compressing each second of audio into 24 measures of instrument dominance, where each of these 24 measures is interpretable through our understanding of psychoacoustics. Furthermore, this approach covers edge cases. For instance, if we calculate which instrument dominates each of the 24 critical bands, and each band is dominated by the guitar, the guitar is obviously too loud unless it's the only instrument playing.

4.1 Measuring Perceptual Instrument Dominance

We calculate perceptual instrument dominance by first taking the Short-Time Fourier Transform (STFT) of each audio stem to measure the amplitude of frequency bins over time. We use one-second windows of audio to make decisions, so we take the STFT with a window size and hop length both equal to the number of samples in a one-second window. We code in the frequency values of each critical band cutoff from Zwicker and iterate through each time step t and each band. For each critical band frequency range, we calculate the **95th percentile amplitude** over all of the audio stems. Then, we count the number of frequencies played by each instrument whose amplitude exceeds the 95th percentile. We normalize these counts by dividing by the total count over all instruments to get the mix (length four array) for that time step and critical band. We finish iterating over the bands and time steps and return the $(t \times 24 \times 4)$ array.

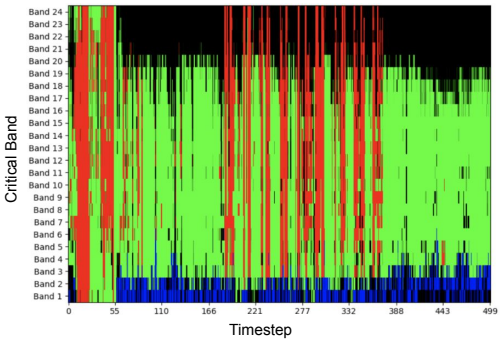

Figure 2: Graphing our perceptual instrument dominance measure over time and band. Green is guitar, red is vocals, blue is bass, and black is drums. If two instruments are equally dominant in the critical band, the color is interpolated.

This approach is a simple approximation of perceptual masking. Generally, audio has a pre and post masking effect depending on the time from the onset of the signal [7]. Additionally, the Zwicker studies used sine tones, so instrument tones may trigger other effects that make them easier to hear. We used our approximation for two reasons. First, we base our work on the demonstrated concept that stronger tones mask weaker tones in critical bands. Second, we can graph t seconds of audio like in **figure 2** and see that the mixing information conveyed is consistent with the general understanding of mixes (the bass is dominant in the low frequencies, the drums are transient, and the guitar is in the mid frequencies). If we include all frequencies, not just dominant frequencies, these clear visual relationships start to fade. We chose the 95th percentile rather than taking the highest amplitude frequency in each band to make the change between time stamps across the same critical band more continuous. P_{95} is a common signal processing threshold, and worked well in our visualizations compared to P_{90}, P_{75}, and so on.

5 Reference Track

A reference track is a music recording that embodies the band's ideal mix and could come from their past recordings, another band, or even a studio recording. The reference track we used for our development and early testing was the "Little Wing" recording from John Mayer in Webster Hall, NYC, 12.28.2004, downloaded from the Live Music Archive (LMA), a popular data source for live audio tasks [2, 19].[3] This John Mayer recording was particularly well recorded and had our expected four-part instrumentation.

We passed the reference track through Demucs to obtain audio files for drums, guitar, bass, and vocals. We noticed two shortcomings with the separated audio. First, some parts of John Mayer's guitar solos are included in the vocal stem. Second, the significant crowd noise (cheering, clapping, etc) that occurs in any professional show is included in the vocal stem. We do not use any metadata in the training process.

6 Finding the Ideal Mix

SoLAR works by finding the difference between the current perceptual audio instrument-wise mix and the ideal mix. We calculate the current perceptual mix through the same process described in **section 3.1**.

Our audio dataset has the ideal perceptual mix, but the mix varies greatly throughout the performance. Naturally, if the musician is not playing, we will not hear them, but that does not give any insight into their volume level. Therefore, we need some measure of which artists are currently playing from their audio signal that tells us whether we should expect them in the mix. Once we have this measure, we can approximate a function to get the mix using a set of small neural networks.

6.1 Who is Playing?

We calculate which instruments are playing in a given window by calculating whether the root mean squared (RMS) of the audio is over some set threshold. RMS is invariant to the length of the audio signal, so we take the RMS of 10 ms subsections to account for any sudden peaks. We set an experimentally derived silence threshold $RMS < 10^{-3}$ from testing with our John Mayer dataset. We calculate who is playing for a one second chunk of audio, so if any of the 10 ms subsections are above the silence threshold, we say that the instrument was playing in the one second chunk. This strategy gives an array of 0/1 of length four, where 0 is silence, and 1 is playing. Therefore, we have $2^4 = 16$ possible playing combinations.

We considered extracting more details from the raw audio stems to find additional intent measures. For instance, a guitarist fits into the mix differently when they are playing chords compared to taking a solo. We experimented with more granular measures, but ultimately those measures contained too much information to effectively map to a mix in our dataset without overfitting.

Another expansion we considered is dynamically finding the threshold for silence. We set a constant value to reduce the system configuration time, but another possible solution is to record each artist at the start of their soundcheck and use that audio to measure the threshold for silence.

6.2 Approximate Mix Mapping Function

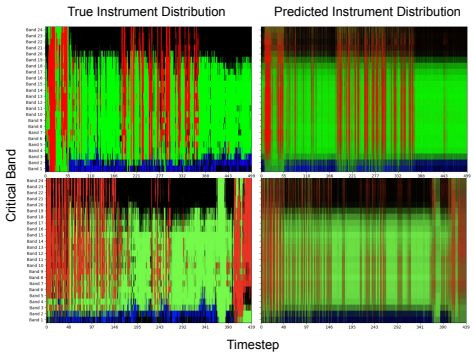

Figure 3: Actual vs predicted mix. The top row is actual (left) vs predicted (right) for 500 seconds of audio in the training set. Bottom row is actual (left) vs predicted (right) for a held-out test audio.

SoLAR uses 24 small neural network models to map our playing measure onto the expected perceptual mix. Each model takes the four silent/playing values and maps them onto an output of four floats in [0, 1] that measure the perceptual dominance share of each instrument in **one** critical band. Each model has one hidden layer with a ReLU activation between the hidden and output layers and a softmax activation from the output layer to enforce that the sum of all perceptual dominance shares should be 1 to represent a percentage. We use 24 small models rather than one large model to reduce parameter sharing, which would create unwanted blurring between critical bands based on our experiments. Please reference **figure 4** for the model diagram.

Each model is trained with mean-squared error loss (MSE) on 10 epochs at a 0.01 learning rate and a batch size of 32. We tested our model with a held-out recording from John Mayer for the same concert. The test MSE diverged slightly from the train MSE on the middle bands, but we observed that the predicted and real perceptual dominance values were visually similar, shown in the bottom image on **figure 3**. Please note that other statistical methods could predict perceptual dominance similarly well; we used our architecture because neural networks reliably learned correct mappings.

[3] https://archive.org/details/jmayer2004-12-28.4011.trak2.flac16/jmayer2004-12-28d1t02.flac

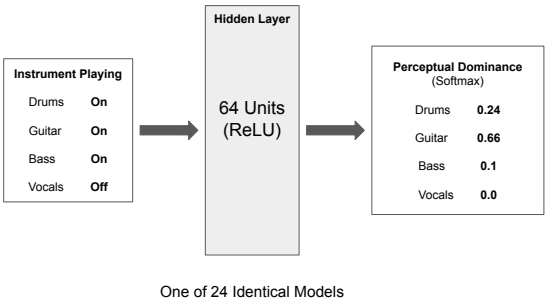

Figure 4: Neural Network architecture for 24 models mapping playing on/off values to reference mix composition. The example model would likely be a mid-frequency band. The model architecture was iteratively refined with reference to the MSE loss and the graphical performance.

7 Adjustment Recommendation

We frame soundchecking as a reinforcement learning problem and make adjustment recommendations with a policy function that maps state onto action $\pi(s) = a$, where our state is the mix and the action is chosen from the set {turn up, turn down, stay} [18]. To get our state mix, we subtract the predicted mix from the actual mix. That is, we take the element-wise difference between the (24×4) actual mix array at the current time, and the (24×4) predicted mix from the model detailed in **section 5.2**.

If an element in the resulting mix state is positive, that means that the instrument dominated that critical band more than expected, with the opposite being true for negative elements. Zero elements mean that our real mix was the same as the predicted, ideal mix. Therefore, we calculate RL reward as proportional to the additive inverse of the absolute valued sum of the mix difference.

Interestingly, we were able to construct a policy function without needing any actual reinforcement *learning*. From inspecting the mixing behavior, if we take the (24×1) array slice that represents the guitar mix (actual - predicted) and take the sum over all critical bands, the following policy works well:

- If the sum is greater than 3, turn down
- If the sum is less than -3, turn up
- Otherwise, stay

We determined the cutoff of 3, −3 through the simulated experiments, discussed below. Please note that values of 4, −4 and 5, −5 also worked similarly well.

7.1 Simulated Experiments

We constructed a gymnasium environment to simulate real mixing practices [3, 20]. Each time the gym is **reset** a 60 second audio section is sampled (with separated stems) and a guitar gain delta is randomized between -60 dB to +60 dB using Pedalboard [16].[4] Imagine the gain delta as "messing up" the guitar volume level. A gain delta of +60 dB would mean increasing the actual (ideal) John Mayer guitar gain by 60 dB. Each **step**, we take the action from the policy function by modifying the guitar gain delta. Then, we get the total mix by adding together all the stems (including the gain-adjusted guitar) and get a reward based on how close we are to zero elements in the mix difference array.

This gym environment is **solved** when the final gain delta is near 0 dB. This solution is non-obvious because the gain delta is not included in the state.

We tested four policy functions:

(1) * Our hard-coded policy function
(2) **(baseline)** A random walk over the action space (for each step, select a random action)
(3) **(baseline)** Always staying at our initial random gain
(4) A learned policy with deep Q-Networks (DQN) [9]

We ran experiments in the gymnasium environment and our hard-coded policy consistently got the highest total rewards over each episode, beating the two baseline functions significantly. We also examined the final guitar gain level at the end of each episode, and the final gain delta was consistently adjusted from [-60, 60] to [-12, 12] dB. Therefore, taking the actions from the policy function were interpretable as fixing the "messed up" gain back to the recorded reference gain. The DQN policy was able to achieve similar rewards with 100 episodes of training, but felt overengineered for our current task. Future work with added live mixing aspects (EQ, reverb, or distortion, for example) could necessitate RL training in our gym environment.

8 Initial Interface

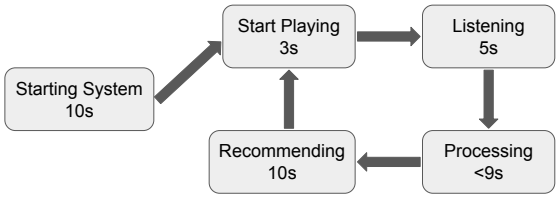

Figure 5: States and timing (in seconds) for initial GUI screens

We created a graphical user interface (GUI) using the PyGame library. This interface, which is started from the terminal, is made to run on a laptop computer in the listening area of a performance space. We record five seconds of audio each iteration because one second is too little data for Demucs. Our first prototype followed the finite state machine in **figure 5** on a timer.

When drafting the interface, we considered user wait times for both Demucs processing time, which can take up to nine seconds on each audio clip, and recording time. In order to give consistent loading-time feedback, we created a loading bar and count-down timer. We also created large visual icons for volume, up, down, and recording, because the user will be far from the computer screen. Please see **figure 6** for example screenshots of these windows.

9 Formative Study

In addition to the first author playing extensively with the system, we set up a small usability study to test that guitarists are able to use this interface and that everything functions as intended.

[4] A standard 40W Marshall guitar amplifier can go up to 100 dB https://marshallforum.com/threads/dsl-40-c-volume-compared-to-dsl-100-h.124482/post-2221747

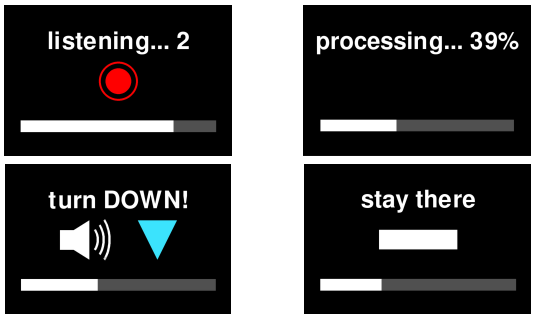

Figure 6: Initial GUI screens. From left to right and top to bottom: listening/recording screen, loading screen for processing pipeline, instruction to turn down amplifier, instruction to not change amplifier. Please note that these screens take up the entire laptop screen and are made to be visible from far away.

We had two participants, each of whom is a proficient guitarist with performance experience. Each participant was asked to send us a reference song before testing with their ideal mix. One participant sent a live recording of "Layla" with Eric Clapton and Derek Trucks, and the other participant asked to soundcheck with John Mayer, so we used the "Little Wing" recording. Additionally, participants were asked to bring their guitar and amplifier.

The room we used is a university music performance space with an audience capacity of 90 people. We simulated performance by removing the guitar stem from the reference song and playing that backing track on the room speakers. Using recorded audio rather than bringing a drummer, bassist, and vocalist makes our test less realistic, but still yielded useful early findings for the design and effectiveness.

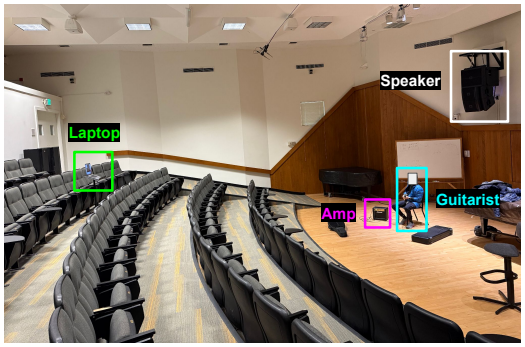

Figure 7: Room setup for SoLAR test. Annotated: MacBook Pro (2019) laptop running SoLAR using internal microphone, Marshall amplifier, (anonymized) guitarist, and speaker playing background track. The SoLAR GUI took up the full screen so the player could clearly see the text and icons (please see figure 6 and figure 9).

We set a laptop computer running the SoLAR interface in the room's listening space. Then, each guitarist followed the instructions on the interface screen. Each guitarist played their guitar to the backing track. Notably, they could improvise their part and were not required to replicate the guitar playing from the reference track. Please see **figure 7** for the room setup. We sat in the listening area and observed how easily the guitarist could read and understand the instructions, how they adapted to the recommendations, how well the timing on the state machine worked, and whether the recommendations seemed appropriate given the sound we could hear. We also asked the waiting guitarist to sit with us for additional listener feedback. The first guitarist performed three soundchecks on "Layla", and the second guitarist performed two soundchecks on "Little Wing".

10 Feedback and Interface Redesign

We interviewed the two guitarists at the same time in a conversation after the study concluded with the following questions:

- (to both the player and the listener) Were the final loudness levels consistent between trials?
- (to both the player and the listener) Did the final loudness levels sound correct for the reference track?
- Would you change anything about the interface?

In response to the first question, both guitarists found the final loudness levels to be consistent between trials, as in soundcheck one for "Layla" resulted in a similar guitar loudness as soundcheck two and three (same for "Little Wing"). In response to the second question, both the player and the listener said that the guitar was too loud on the "Layla" soundcheck, but that the guitar was the right loudness on the "Little Wing" soundcheck. We attribute the excessive loudness in "Layla" to the fact that the live recording has two guitarists, and Demucs groups both guitars into the "other" category, creating a much louder sound than one guitar.

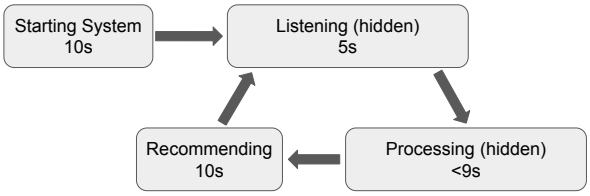

Figure 8: States and timing (in seconds) for redesigned GUI screens

When asked about changes to the interface, both guitarists advocated for an interface redesign. Both guitarists reported that having too much feedback, both in the form of text and a loading bar, was distracting when they were trying to play. Both guitarists agreed that they did not need a screen to tell them to start playing or that the system was "listening" because they would be playing their guitars by default. Therefore, both guitarists suggested a system that only displayed information when there was an actionable recommendation (turn up or turn down) and a blank screen otherwise.

We redesigned the interface to remove the unnecessary visual information. Please see **figure 8** for the updated FSM and **figure 9** for a sample of the updated GUI screens. We showed the redesigned interface to the two study participants, and both participants agreed that the new interface incorporated their feedback from the usability study and improved on the previous design.

Figure 9: Redesigned GUI screens. From left to right and top to bottom: get in position, listening/processing (hidden), instruction to turn up amplifier. Please note that these screens and icons are made large to be visible from far away.

11 Conclusion

We introduced SoLAR, a system that gives mixing recommendations directly to a guitarist. We demonstrated that SoLAR's recommendations work in a simulated environment by both returning an adjusted guitar gain to the reference gain and outperforming two baseline recommendation policies. We also found promising results in a small-sample live test. We followed an iterative interface design process and found that a condensed system was preferred when designing our GUI.

Future work on SoLAR will involve scaling to more instruments/effects and conducting a larger user study with a full band, both to find the general effectiveness of the mixing recommendations and to iterate further on the GUI design. Improvements to SoLAR could focus on more extensive testing with capabilities of a laptop microphone, more study participants, scenarios where the reference track differs from the performance track, and runtime efficiency improvements in source separation.

12 Ethical Standards

We obtained IRB approval for our usability study and obtained informed consent from all participants. We needed to ensure that participants were proficient guitar players to properly utilize SoLAR, so we selected from people the first author personally knew who were skilled at guitar. In order to combat bias, the first author was careful not to show the interface to participants before the study. We avoided questions that would clearly introduce bias (such as "would you use this").

References

[1] Yehiel H. Amo, Gil Zissu, Shaltiel Elul, Eran Shlomi, Dima Schukin, and Almog Kalifa. 2014. Managing Live Music Bands via Laptops using Max/MSP. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*. Goldsmiths, University of London, United Kingdom.

[2] Sean Bechhofer, Simon Dixon, György Fazekas, Thomas Wilmering, and Kevin R. Page. 2014. Computational Analysis of the Live Music Archive. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. Taipei, Taiwan.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540* (2016).

[4] Michael Clemens and Ana Marasović. 2025. MixAssist: An Audio-Language Dataset for Co-Creative AI Assistance in Music Mixing. *CoRR* abs/2507.06329 (2025). Published at the Conference on Language, Music, and Multi-Modal Interaction (COLM 2025).

[5] Alexandre Défossez. 2021. Hybrid Spectrogram and Waveform Source Separation. In *Proceedings of the ISMIR 2021 Workshop on Music Source Separation.*

[6] Rebecca Fiebrink, Dan Trueman, and Perry R. Cook. 2009. A Meta-Instrument for Interactive, On-the-Fly Machine Learning. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*. Pittsburgh, PA, United States, 280–285. https://doi.org/10.5281/zenodo.1177513

[7] Jürgen Herre, Schuyler Quackenbush, Minje Kim, and Jan Skoglund. 2025. Perceptual Audio Coding: A 40-Year Historical Perspective. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Hyderabad, India, 1–5. https://doi.org/10.1109/ICASSP49660.2025.10887760

[8] Davide Lionetti, Luca Turchet, Massimiliano Zanoni, and Paolo Belluco. 2024. Muscle-Guided Guitar Pedalboard: Exploring Interaction Strategies Through Surface Electromyography and Deep Learning. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*. Utrecht, The Netherlands, 241–251. https://doi.org/10.5281/zenodo.13904842

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. In *Proceedings of the 2013 Neural Information Processing Systems (NeurIPS) Deep Learning Workshop*. Lake Tahoe, NV, USA.

[10] David Moffat and Mark B. Sandler. 2019. Automatic Mixing Level Balancing Enhanced Through Source Interference Identification. In *Proceedings of the 146th Audio Engineering Society Convention (AES)*. Dublin, Ireland. E-Brief 497.

[11] Enrique Perez-Gonzalez and Joshua D. Reiss. 2009. Automatic Gain and Fader Control for Live Mixing. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz, NY, USA, 1–4. https://doi.org/10.1109/ASPAA.2009.5346498

[12] Darius Petermann, Gordon Wichern, Aswin Subramanian, and Jonathan Le Roux. 2023. Hyperbolic Audio Source Separation. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[13] Eric Roberts. n.d.. *MP3: The Concept of MP3 Compression.* Stanford University Computer Science. https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossy/mp3/concept.htm

[14] Simon Rouard, Francisco Massa, and Alexandre Défossez. 2023. Hybrid Transformers for Music Source Separation. In *ICASSP 23*.

[15] Bowen Shi, Andros Tjandra, John Hoffman, Helin Wang, Yi-Chiao Wu, Luya Gao, Julius Richter, Matt Le, Apoorv Vyas, Sanyuan Chen, Christoph Feichtenhofer, Piotr Dollár, Wei-Ning Hsu, and Ann Lee. 2025. SAM Audio: Segment Anything in Audio. (2025). https://arxiv.org/abs/2512.18099

[16] Peter Sobot. 2021. *Pedalboard.* https://doi.org/10.5281/zenodo.7817838

[17] Christian J. Steinmetz, S. S. Vanka, M. A. Martínez Ramírez, and G. Bromham. 2022. Deep Learning for Automatic Mixing. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*.

[18] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press, Cambridge, MA, USA.

[19] Florian Thalmann, Eita Nakamura, and Kazuyoshi Yoshii. 2022. Tracking the Evolution of a Band's Live Performances over Decades. In *Proceedings of the 23rd International Society for Music Information Retrieval Conference (ISMIR)*. International Society for Music Information Retrieval, Bengaluru, India, 850–857.

[20] Mark Towers, Ariel Kwiatkowski, Jordan K. Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. 2025. Gymnasium: A Standard Interface for Reinforcement Learning Environments. In *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS 2025), Spotlight Poster*. San Diego, CA, USA.

[21] Augoustinos Tsiros and Alessandro Palladini. 2020. Towards a Human-Centric Design Framework for AI Assisted Music Production. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*. Birmingham, UK. https://doi.org/10.5281/zenodo.4813436

[22] Soumya Sai Vanka, Maryam Safi, Jean-Baptiste Rolland, and György Fazekas. 2023. Adoption of AI Technology in the Music Mixing Workflow: An Investigation. In *Proceedings of the 154th Audio Engineering Society (AES) Europe Convention*. Helsinki, Finland. https://doi.org/10.48550/arXiv.2304.03407

[23] Soumya Sai Vanka, Maryam Safi, Jean-Baptiste Rolland, and György Fazekas. 2024. The Role of Communication and Reference Songs in the Mixing Process: Insights from Professional Mix Engineers. *Journal of the Audio Engineering Society* 72, 1/2 (2024), 5–15. https://doi.org/10.17743/jaes.2022.0123

[24] Duncan A.H. Williams. 2020. MINDMIX: Mapping of Brain Activity to Congruent Audio Mixing Features. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*. Birmingham, UK, 349–352. https://doi.org/10.5281/zenodo.4813408

[25] Eberhard Zwicker, G. Flottorp, and S. S. Stevens. 1957. Critical Band Width in Loudness Summation. *The Journal of the Acoustical Society of America* 29, 5 (1957), 548–557. https://doi.org/10.1121/1.1908963