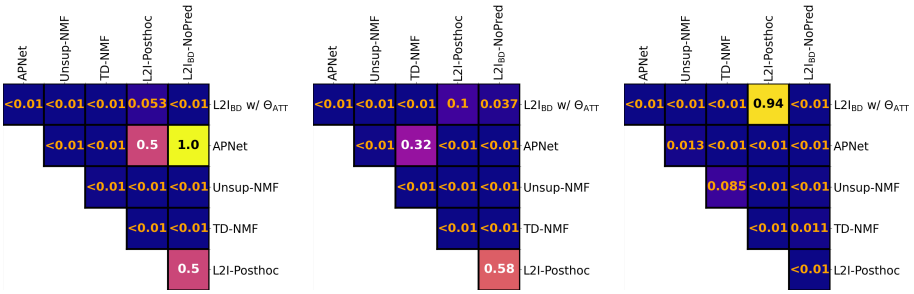

Figure 1: ESC50 (Left), SONYC-UST (Middle), OpenMIC-2018 (Right) p-value for classification performance of all pairs of by-design interpretable systems

1 Significance test results

We conducted a significance test for the by-design interpretable networks in Tab. VIII (main paper) and fidelity methods in Tab. IV (main paper). In each case, we compared the samples of observed performance/fidelity scores for each pair of systems via a student T-test under the assumption of identical population variances. The null hypothesis was that the two independent sets of performances have identical expected value. In the figures below we report the p-values for the significance test for each dataset.

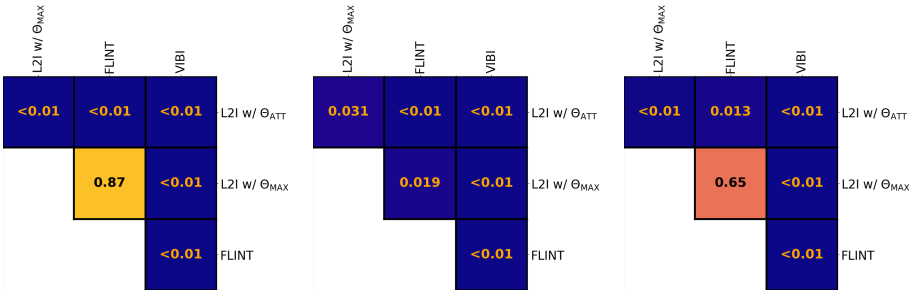

Figure 2: ESC50 (Left), SONYC-UST (Middle), OpenMIC-2018 (Right) p-value for fidelity performance of all pairs of systems

We didn't conduct such analysis for performances in Tab. I since our goal in reporting the results was to indicate that the model we interpret for post-hoc interpretation is a complex system with strong performance.

2 Architecture modification for OpenMIC

Given the different nature of OpenMIC, upon initial experiments, we observed that interpretations of multiple classes for a sample would often come from a

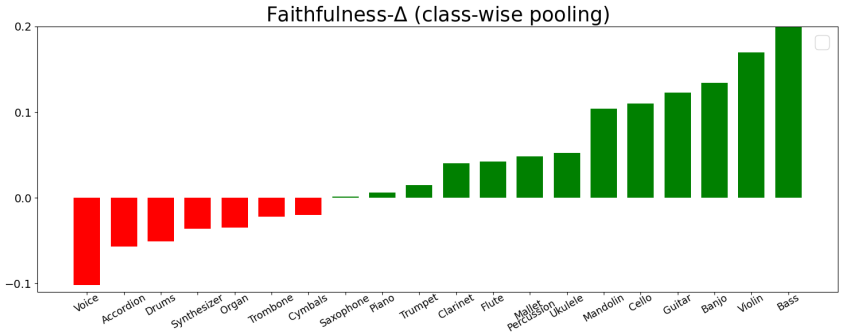

Figure 3: Difference in faithfulness $\text{FF}_{\text{median}}$ on OpenMIC classes for class-wise attention modification. Classes in red denote drop in median faithfulness after modification while those in green denote increase. Only 'Voice' class shows a strong drop while for many classes faithfulness improves strongly. Moreover, for 5 out of 6 classes (including voice) with drop in faithfulness, the updated model still has a positive faithfulness.

similar set of components. In other words, interpretations for different classes would attend to very similar components and input time frames, thus sounding very similar. To alleviate this issue we made the architectural modification and allowed class-wise pooling for intermediate embedding $\mathbf{H}_{\mathcal{I}}(x)$ by generating a separate attention vector for each class. This helped improve the problem as the prediction for each class could be made separately by attending to different time frames and emphasizing different components. Quantitative effect of this modification is indicated in figure 3. For most classes adding this modification positively impacts faithfulness. For only class 'Voice', we observed a noticeable negative impact. Overall, it improves faithfulness significantly as well as improves fidelity by a small extent (mean 0.908 compared to 0.920 after modification).

3 Baseline implementations details

FLINT: We implemented it with the help of their official implementation available on GitHub.[1] For each experiment, we fix their number of attributes J equal to the number of our NMF components K. We also choose the same hidden layers for their system as we choose for ours. This baseline is trained for the same number of epochs as us. We use same values for our $\mathcal{L}_{\text{NMF}}$ loss weight, α, and their $\mathcal{L}_{if}$ loss weight γ. For the other loss hyperparameters, we use their default values and training strategy.

VIBI: We implemented this using their official repository.[2] The key hyperparameters that we set are the input chunk size and their parameter K, the

[1] `https://github.com/jayneelparekh/FLINT`
[2] `https://github.com/SeojinBang/VIBI`

number of chunks to use for interpretation. We use a larger chunk size than in their experiments to limit the number of chunks. On ESC-50, we use a chunk size of 32×43, and on SONYC-UST, a chunk size of 32×86. This yields 40 chunks for each input on both the datasets. We varied the K from 5 to 20, and report the results with best fidelity. The system was trained for 100 epochs on ESC-50 and 30 epochs on SONYC-UST

SLIME: We primarily relied on implementation from their robustness analysis repository [3]. The key hyperparameters to balance are the number of chunks vs chunk size. SONYC-UST contains 10 second audio files. This is much longer than 1.6 second audio files for which SLIME was originally demonstrated [1]. Therefore, we divide only on the time-axis to limit the number of chunks. SLIME recommends a chunk size of at least 100ms. They operate on upto 290ms chunk size. We balance these two hyperparameters by dividing our audio files in 20 chunks of 500ms chunk size. We select a maximum of 5 chunks for interpretations and a neighbourhood size of 1000.

APNet: We utilized their source code [4] for implementing their method on our datasets. We did not modify their network design or loss weights and set the number of prototypes same as our number of components. The number of mel filters was chosen between 64 and 128. We trained their system for 100 epochs on ESC-50 (each fold) and 21 epochs for SONYC-UST and OpenMIC-2018 and report the highest recorded metrics.

NMF variants: For implementing both TDL-NMF and Unsupervised-NMF, we utilized the source repository [5] of TDL-NMF. The unsupervised-NMF variant simply trains a linear model on top of generated time activations for predictions while the dictionary is also updated with classification loss for TDL-NMF. We trained dictionaries of multiple sizes, ranging from 32 to 256 for each dataset and two different audio representations, log-magnitude spectrogram and mel-spectrogram. The best performance among all these configurations is reported.

4 Subjective evaluation implementation

The subjective evaluation interface was implemented using webMUSHRA [2]. Prior to voting on the test samples, participants were provided with an instruction page and then a training page with an example to get used to interface, instructions, tune their volume etc. Screenshots of the instruction and training page are given in Fig. 4, Fig. 5 respectively.

References

[1] S. Mishra, B. L. Sturm, and S. Dixon, "Local interpretable model-agnostic explanations for music content analysis." in *ISMIR*, 2017, pp. 537–543.

[3] `https://github.com/saum25/local_exp_robustness`

[4] `https://github.com/pzinemanas/APNet`

[5] `https://github.com/rserizel/TGNMF`

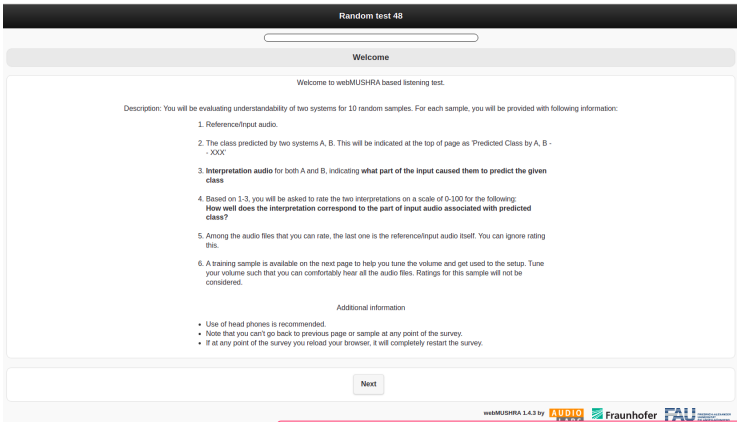

Figure 4: Instructions for the participants at the start of the subjective evaluation

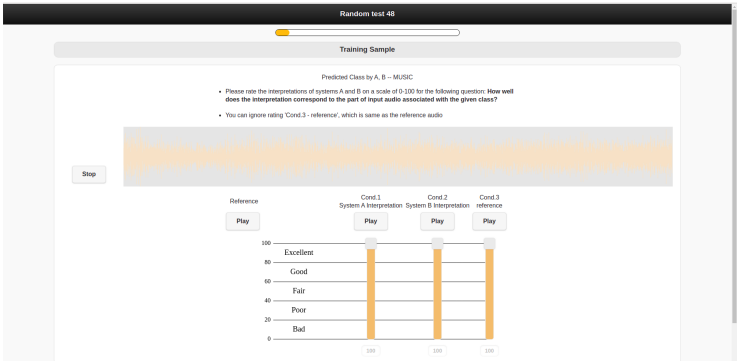

Figure 5: Training page for subjective evaluation that illustrates the interface for scoring for the participants.

[2] M. Schoeffler, S. Bartoschek, F.-R. Stöter, M. Roess, S. Westphal, B. Edler, and J. Herre, "webmushra—a comprehensive framework for web-based listening tests," *Journal of Open Research Software*, vol. 6, no. 1, 2018.