Adversarial Noise Generation to Disguise AI-Generated Music: A CLAP-Based Approach

Aqsa Kausar, Dino Hamidovic, Georgios Vidalakis, Ankit Grover

Abstract

This paper investigates the vulnerability of AI music classifiers to adversarial attacks, focusing on the ability to manipulate AI-generated music to bypass detection and classification. We utilise CLAP embeddings for feature extraction and train a Multilayer Perceptron classifier to distinguish between human made and AI-generated music. In order to deceive this classifier, we generate adversarial noise that is imperceptible to human listeners but can trick the model into misclassifying AI-generated music as human made. This is achieved through an iterative optimization process that involves adding noise to the audio signal and refining it using back propagation. We carefully control the Signal to Noise ratio(SNR) to ensure the noise remains inaudible while still effectively fooling the classifier. Our experiments demonstrate that adversarial noise generation can jeopardize the reliability of AI music classification systems, highlighting the need for improved security measures and the development of more resilient AI systems in the field of music classification.

1 Introduction

This project investigates the robustness of AI-generated and human made music classification methods. Firstly, we will use CLAP embeddings from AI and human generated music. Then we will train different models on these embeddings to see which models perform the best at classification. Then, we create a method that can add adversarial noise to AI generated audio clips so that the trained model miss-classifies it as human generated music.

According to previous research on AI and human music, entropy provides a valuable lens through which to examine the fundamental differences between human and AI music (Ren, I. Y., (2015). In other words, AI generated music tends to be more predictable than Human made music. This suggests that entropy could be a weakness users could exploit in adversarial attacks - AI music could be modified to sound more unpredictable by adding unusual note sequences or more unique sound selection. However, it's infeasible to directly modify individual notes in AI-generated audio clips to make them sound more human. Such changes would likely be perceptible to the human ear and could introduce artifacts that compromise the naturalness of the music. Therefore, adversarial noise generation, also based on the concept of entropy, emerges as a more effective strategy. By carefully introducing subtle, imperceptible perturbations to the audio signal, we can manipulate the entropy of the AI-generated music, making it appear more unpredictable and human-like. This approach aims to exploit the inherent limitations of AI music classifiers and deceive them to misclassify AI-generated music as human-made.

A similar method has been used in previous research by Christian Szegedy et al. (2014) in the field of image recognition. Their work demonstrated that even small, imperceptible changes to images could significantly impact the predictions of neural networks. Other work, By Subramanian et al. (2019), showed that similar principles can be applied to the domain of music. In their work they used white noise and the fast gradient sign method to trick a sound event classifier. They also explained that they were able to generate successful adversarial attacks with high confidence and low perturbation. Finally, other work by Saadatpanah (et al. 2020) has shown that copyright detection systems that are based on machine learning are very susceptible to adversarial attacks. This suggests that by carefully introducing noise to AI-generated music, it may be possible to create adversarial examples that can fool classifiers and challenge the robustness of AI in music applications.[1][2][4]

Adversarial attacks can reveal vulnerabilities in the classifier's design and implementation, and Research suggests that adversarial attacks are becoming a serious security issue in in audio signal processing systems

that leverage machine learning (Duan et al., 2022). Adversarial attacks on speech-to-text systems can result in incorrect transcriptions, causing the system to misunderstand spoken input. In voice assistants, adversarial noise can trick devices like Alexa or Siri into executing unintended actions. Similarly, in audio classification systems used for tasks such as emotion recognition or environmental sound detection, adversarial attacks can lead to misclassification, potentially resulting in safety concerns in critical applications.[3]

By understanding how these attacks work, researchers can improve the robustness and reliability of AI music classifiers, making them more resistant to manipulation and ensuring their fair and equitable use. If a classifier is easily fooled by adversarial attacks, it could lead to unfair or biased outcomes in applications such as music recommendation or copyright infringement detection. For instance, an inaccurate classifier could unfairly promote or demote music based on its perceived origin, rather than its quality.

Difference of AI vs human music, importance of this. For instance, if people can bypass classifiers that will be implemented online Definition of adversarial attacks, examples

2 Method

2.1 Data

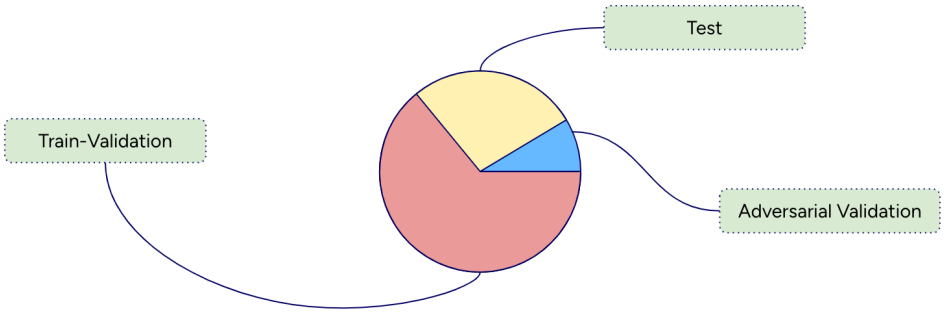


Figure 1: Data split for training, validation, and testing sets.

We employed a subset of the Rolling udio dataset. This collection comprises 500 widely recognized human-composed songs and 500 AI-generated tracks. To evaluate the performance of different models, we partitioned this data into training, validation, and testing sets. Regarding the adversarial attacks, we needed separate validation and test data. . AI generated music was made through Udio. Around half of the songs were generated on the first version of Udio and the other half on version 1.5

2.2 Feature Extraction

We use embeddings extracted from CLAP for both AI generated and human made music datasets, as our features. CLAP is a deep learning model specifically designed for audio representation. It is trained on a large dataset of audio and learns to extract meaningful features from the audio signal (Saijo, et al., 2024). The paper by Benjamin Elizalde et al. (2022) demonstrates the effectiveness of CLAP embeddings for various downstream tasks, including sound event classification, music genre recognition, and speech-related tasks. CLAP consistently outperforms other state-of-the-art models, showcasing its ability to capture meaningful audio features and generalize to new tasks.[5]

CLAP Model

CLAP (Contrastive Language-Audio Pre-training) is a self-supervised model trained on large-scale audio and text pairs to learn versatile audio representations. CLAP was trained on millions of audio samples, encompassing a vast range of sounds and categories, including speech, music, environmental sounds, and more. This extensive dataset, sourced from various online repositories, enables CLAP to learn generalizable audio features applicable to diverse tasks.[6]

CLAP's architecture is similar to other contrastive pre-training frameworks, with separate encoders for audio and text. The audio encoder typically leverages architectures like Transformers or CNN-based models

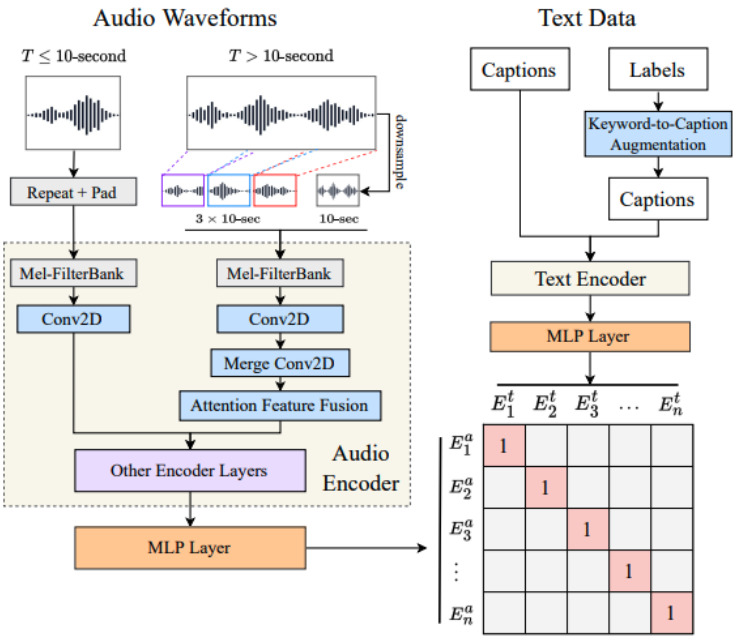


Figure 2: CLAP Model Architecture

to capture temporal and frequency information in audio data, while the text encoder (often a Transformer) processes corresponding textual descriptions. The two encoders are jointly trained in a contrastive learning framework, where positive pairs (matching audio-text pairs) are pulled closer in the embedding space, and negative pairs are pushed apart. This training objective allows CLAP to learn a shared embedding space for audio and text, making it effective at capturing semantic information.

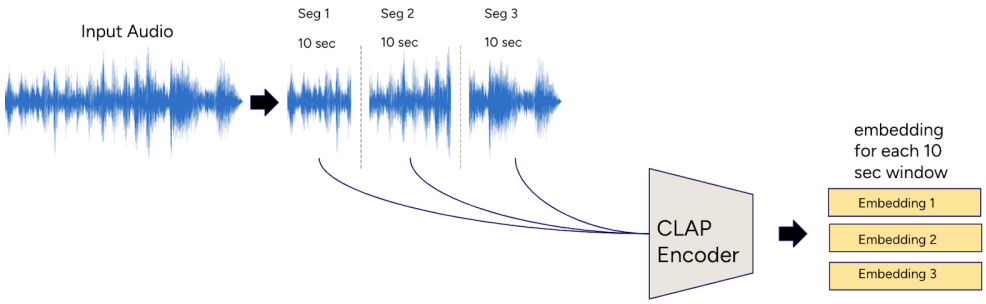
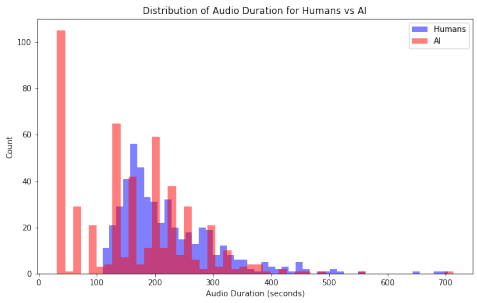


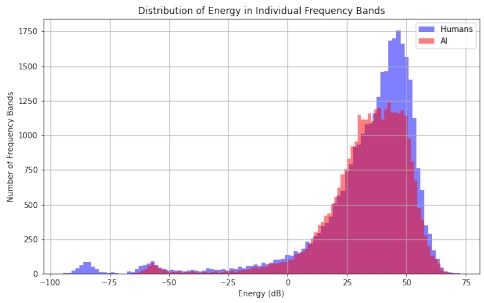
Figure 3: Embedding Extraction using CLAP Encoder

CLAP embeddings are particularly valuable for downstream audio tasks because they capture high-level semantic information, bridging the gap between human language and audio features. These embeddings encapsulate not just the acoustic characteristics but also contextual meaning, enabling nuanced representations. This makes CLAP effective for applications like genre classification, sentiment analysis, sound event detection, and distinguishing between AI-generated and human-created music. Its broad training data and contrastive objective provide a robust foundation for tasks requiring semantic understanding of audio, leading to embeddings that are both descriptive and generalizable. By extracting embeddings from CLAP, we leverage these rich, semantically-aligned representations, which provide a strong starting point for our classification task between Human and AI.

2.3 Exploratory Data Analysis



(a) Distribution of track durations



(b) Energy distribution in all frequencies

Figure 4: Comparison of track duration and energy distribution

We analyzed our dataset before training by extracting Mel Spectrum features using the same sampling rate of 48KHz and window length as used in intermediate CLAP embeddings extraction. We notice that, probably due to compute complexity associated with AI music generation, the majority of the audio lengths are short. The Figure above demonstrates the given phenomena, showcasing an almost bi-modal distribution in audio-lengths for AI generated music compared to uni-modal distribution for humans composed music. Further, this gives us insight into the fact that audio-lengths might be a factor that the model could potentially learn for classifying. In the b) we notice that due to audio-lengths the peak associated with Kurtosis of the histogram representing frequencies with high energy is noticeably higher in human generated music compared to AI generated music. This may be a result of not only audio-length, but also inconsistencies in reconstructing human speech signals, drums, and other instruments with clarity which was observed qualitatively by listening to samples in our dataset.

Training of Baselines

We conducted a sanity check on our dataset by training several baseline models and comparing their accuracies across the training, validation, and test sets. The classifiers tested included Logistic Regression, K-Nearest Neighbors, Support Vector Classifier, Gaussian Naive Bayes, Random Forest Classifier, and Multi-Layer Perceptron (MLP). From the results, we observed that the MLP model achieved the highest performance on the test dataset, making it the most suitable candidate for further experimentation. As a result, we selected the MLP model as our primary starting point for the adversarial attack experiments.

Model	Train Acc (%)	Val Acc (%)	Test Acc (%)
Logistic Regression	100	95.5	94.5
K-Neighbors Classifier	96.1	88.5	90.0
SVC	98.61	94.0	94.0
Gaussian NB	91.09	84.0	87.5
Random Forest Classifier	100	87.5	91.0
MLP	100	95.5	94.5

Table 1: Classification Accuracies for baseline Models without author-based partitioning

Initially, we didn't separate authors across training and testing datasets, which can lead to classification bias due to the model's potential pre-exposure to an author's unique musical style. After this initial test, both Logistic Regression and MLP achieved a test accuracy of 94.5%. To remove this bias, we then separated authors between the training and test sets to ensure that music styles from the test authors were entirely unseen during training. This separation method refers to stratification.

Stratification was important in this context because it helps prevent bias that could skew the model's performance, especially in scenarios where class distributions may differ between the train, validation, and test sets. In our case, we paid particular attention to the authorship of the music tracks. By ensuring

Model	Train Acc (%)	Val Acc (%)	Test Acc (%)
Logistic Regression	100	92.0	91.5
K-Neighbors Classifier	96.1	77.0	80.5
SVC	98.61	78.5	83.5
Gaussian NB	91.09	82.5	86.5
Random Forest Classifier	100	68.5	76.5
MLP	100	90.5	94.0

Table 2: Classification Accuracies for baseline Models (Author based partitioning Considered)

that music from the same authors did not appear in multiple splits (train, validation, and test), we avoided potential data leakage, where the model could learn artist-specific features rather than generalizable patterns. This careful stratification process was crucial in producing more reliable, unbiased, and generalizable results from our experiments.

Following this adjustment, Logistic Regression maintained an accuracy of 94.5%, while MLP achieved a slightly higher accuracy of 95%. Both classifiers outperformed others in accuracy. Ultimately, we selected MLP, implemented in PyTorch, as our primary classifier due to its high test accuracy and compatibility with gradient-based optimization, which is essential for our downstream adversarial noise generation.

2.4 Model Architecture

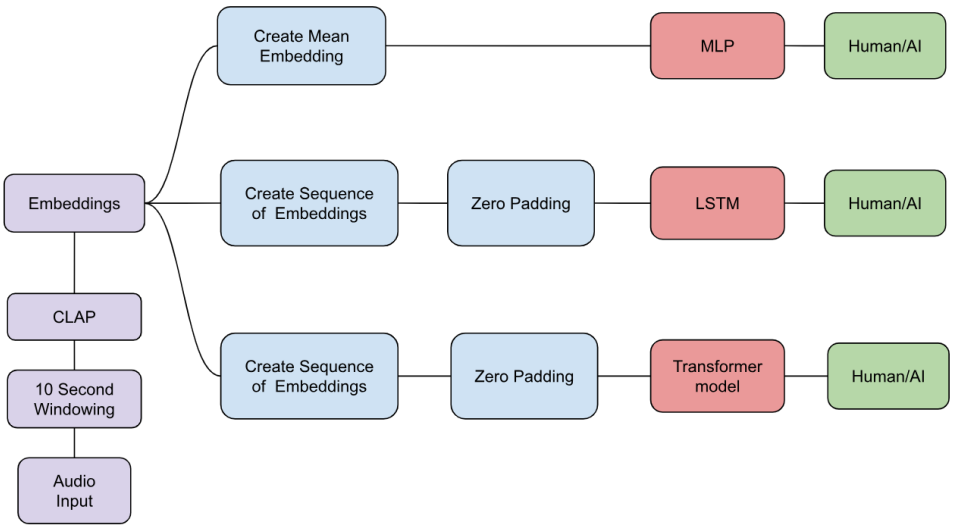


Figure 5: Model Training pipeline for the three different models used

2.5 Model Training

We employed an MLP as our classifier. The MLP consisted of multiple layers, each with a specified number of neurons. To train the model, we utilized the Adam optimizer, which is efficient and has a combination of adaptive learning rates. We set the initial learning rate to 0.001, which controls the step size during parameter updates. We trained the model for 20 epochs, iterating through the training data multiple times. Each epoch involved dividing the data into mini-batches of size 32, ensuring efficient processing and reducing computational overhead. We applied shuffling within each epoch to prevent the model from over-fitting to the training data. We use Binary Cross Entropy loss to train our classifier as minimizing the cross-entropy loss

for classification equates to maximizing the mutual information gained about the two label's distributions thus indirectly minimizing the learn't KL Divergence between the predicted probability distributions and the true distributions.Throughout training, we monitored the training and validation loss to assess the model's progress. We also calculated training and validation accuracy to track the model's performance on both seen and unseen data.

Hyperparams	MLP	LSTM	Transformer
Fit Scaler	No	No	No (LayerNorm)
Hidden Size	64	128	1,024
No of Layers	2	2	3 (pre & post Linear layers)
Dropout	33.6%	38.5%	10.0%
Batch Size	16	16	16
Learning Rate	0.010	0.00063	0.00013
Weight Decay	0.000076	0.000019	0.0047
Bidirectional	No	No	No

Table 3: Hyperparameter Settings for Different Models

Additionally, we computed the F1 score and AUC Area Under the Curve to evaluate the model's ability to discriminate between AI and human-generated music. We visualized the training and validation loss curves to identify trends and potential issues like over-fitting or under-fitting. This visualization helped us assess the model's learning progress and make adjustments if necessary. In order to prevent for over-fitting on the validation set K-Fold cross validation with 7 splits was used. By carefully considering these aspects of the model training process, we aimed to optimize the classifier's performance and ensure its ability to accurately distinguish between AI and human-generated music.

2.6 Evaluation

To evaluate the performance of our MLP classifier, we employed several metrics: Overall Accuracy, F1 score, confusion matrix, and AUC. The overall accuracy score indicates the overall correctness of the model's predictions. A high accuracy suggests that the model is performing well in distinguishing between AI and human-generated music. The F1 score provides a balanced measure of the model's precision and recall. A high F1-score indicates that the model is effectively identifying both AI and human-generated music without excessive false positives or false negatives. The confusion matrix helps to visualize the model's performance in terms of correctly classifying each class. By examining the diagonal elements, we can assess the accuracy for each class, and the off-diagonal elements reveal the types of errors made by the model. Lastly, The AUC measures the model's ability to discriminate between AI and human-generated music. A high AUC indicates that the model can effectively rank instances according to their likelihood of belonging to each class.

Model	CV Accuracy (mean $\pm$ std)	Test Accuracy	Test F1 Score	Test AUC
MLP	97.14% $\pm$ 1.28%	96.50%	96.52%	96.50%
LSTM	96.86% $\pm$ 2.42%	96.50%	96.55%	96.50%
Transformer	97.29% $\pm$ 1.83%	94.50%	94.58%	94.50%

Table 4: Model Performance Metrics

We observe that all our models achieve pretty high scores (greater than 94% test accuracy). With the Transformer being our best model based on cross-validation and MLP and LSTM based on the test set. We can conclude that the AI generated audio, at least in this dataset, contains some characteristics that make it easy to distinguish.

2.7 Adversarial Attack

Adversarial attacks involve manipulating inputs in subtle ways to deceive a machine learning model. These changes are often imperceptible to humans but can cause the model to behave incorrectly. For instance,

small modifications to a single pixel in an image can lead to the model misclassifying it. There are two main types of adversarial attacks: white-box attacks, where the attacker has complete knowledge of the model's architecture and parameters, and black-box attacks, where the attacker lacks direct knowledge of the model but can query it to generate adversarial inputs. The primary goal of adversarial attacks is to cause the model to make incorrect predictions, particularly in sensitive applications such as security systems, facial recognition, self-driving cars, and fraud detection.

Adversarial attacks on audio involve adding small, imperceptible noise to an audio file in order to make a machine learning model misinterpret it. For example, adding such noise to a voice command in a speech recognition system could result in incorrect transcription or unintended actions, such as mistaking "play music" for "delete files". These attacks pose unique challenges because audio is processed differently from images, unfolding over time and requiring consideration of temporal dependencies. Additionally, the adversarial noise must be carefully crafted to avoid noticeable distortion while still causing the model to make errors.

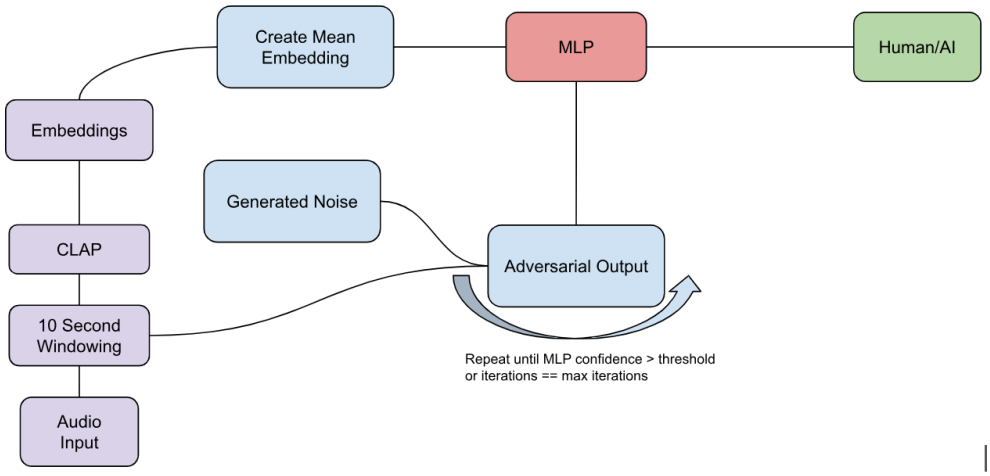


Figure 6: Adversarial attack flow for MLP

The diagram illustrates the process of generating adversarial attacks on an MLP model using music audio data passed through CLAP for embeddings. The audio input is first divided into 10-second windows, and each window is processed by the CLAP model to produce embeddings. These embeddings are used to create a mean embedding, which serves as the input to the MLP classifier. Noise is generated using gradients and added iteratively to the embeddings, with the adversarial output evaluated by the MLP. The process is repeated until the MLP's confidence exceeds a threshold or a maximum number of iterations is reached. The final goal is to classify the AI generated audio as Human generated while introducing adversarial noise to cause misclassification.

$$\text{new audio signal} = \text{old audio signal} - \text{learning rate} \times \text{gradient} \tag{1}$$

We also monitor the Signal-to-Noise Ratio (SNR) during the adversarial attack process to ensure that the generated adversarial audio remains realistic. Specifically, we limit the SNR to a predefined threshold. If the SNR of the adversarial audio exceeds this threshold, the difference is used to adjust the added noise accordingly. This step is critical for maintaining the perceptual quality of the audio, as it ensures that the adversarial noise remains imperceptible to humans while still being effective at fooling the model.

Here, the learning rate controls the size of the adjustment, balancing effective changes without excessive distortion. To ensure the adversarial noise remains inaudible to humans, we maintain a high signal-to-noise ratio (SNR) throughout. Typically, SNRs above 60 dB are considered imperceptible, while those dropping to around 20-30 dB become noticeable. We set an SNR threshold (e.g., 60 dB) and iteratively adjust the noise to stay within this limit. This SNR threshold compliance guarantees that the modifications are subtle enough to be masked by the original audio, while still tricking the classifier.

Additionally, we utilize Projected Gradient Descent (PGD), a method that iteratively refines the noise by projecting the updated signal back onto a feasible set after each gradient step. This keeps the noise level within our SNR constraints, ensuring effective yet inaudible adversarial modifications. By experimenting with different confidence levels (e.g., 0.5 and 0.9) and SNR thresholds, we optimize the balance between stealthiness and classifier confusion. To employ PGD, we define an effective method to compute the projection of a noisy audio signal to the subspace that respects the SNR constraint. Firstly, we observe that setting a lower limit to the SNR is equivalent to setting an upper limit to the power of the noise signal. Since the power of the noise signal is the scaled sum of squares of the noise signal values for each frame, the subspace that respects the constraint is a hypersphere, with its center corresponding to a zero noise signal. If a noise signal represented on this space is inside this hypersphere, then there is no need for a projection. If it is outside of the hypersphere, then the noise power constraint is violated and the projection is done by computing the point on the surface of the hypersphere that is closest to the non-projected point. This projected point is a scalar multiple of the non-projected point. This means that the projection can be achieved by computing the scalar by which we can multiply the noise signal to limit its power to the allowed level, which can be done in time complexity that scales linearly with the number of the signal's frames. By utilizing this projection method, we manage to perform fast PGD iterations.

2.7.1 Attack Evaluation

We evaluate different adversarial attack settings using the adversarial validation split.. We experiment with different parameters of the adversarial attack method and analyze how they affect the adversarial attack outcome. Specifically, these parameters are the learning rate and the minimum allowed SNR, which are used by the projected gradient descent algorithm to update the audio signal, as well as the minimum required confidence (that the manipulated audio is human generated) of the classifier, which acts as a termination condition for the gradient descent algorithm. In all our experiments we set the maximum number of iterations for projected gradient descent equal to 50.

Evaluate a set of parameters as described above on two metrics. The first metric is the attack success rate, which is defined as the percentage of attacks that reached the required minimum confidence. The second metric is the average number of iterations until success which takes into account only the successful attacks. We comprehensively evaluate the hyperparameter-tuned MLP classifier with the highest cross-validation accuracy. The evaluation results are shown in the table below.

Min SNR (dB)	Learning Rate	Success Rate (50% confidence)	Avg # Iterations (50% confidence)	Success Rate (90% confidence)	Avg # Iterations (90% confidence)
None	1e-6	82.98%	20.77	82.98%	21.62
None	1e-5	95.74%	16.87	95.74%	17.51
None	1e-4	97.87%	12.67	97.87%	13.09
50	1e-6	80.85%	20.03	80.85%	21.03
50	1e-5	95.74%	17.09	95.74%	17.71
50	1e-4	97.87%	14.50	97.87%	15.00
60	1e-6	82.98%	20.62	82.98%	21.64
60	1e-5	95.74%	17.73	95.74%	18.38
60	1e-4	72.34%	19.53	72.34%	20.41

Table 5: Adversarial Attack Results for Different SNR Levels and Learning Rates

By observing these results, we notice that as the minimum required confidence increases, the success rate is consistently getting lower and the average number of iterations higher. This is expected, because it is more difficult to make the classifier misclassify an audio with higher confidence. We can also notice that, for most cases, increases of the learning rate translate to increases of the success rate and decreases of the average number of iterations. This is because the smaller learning rates (1e-6 and 1e-5) are quite small for this task and the adversary converges to a solution slowly, often reaching the maximum number of iterations.

Finally, we notice that when the minimum SNR is high (60dB), a high learning (1e-4) rate can be problematic. This happens because the high learning rate causes the minimum SNR to be reached quickly, without considerable increases of the confidence, and then the projection mechanism makes further increases of the confidence slower or impossible, depending on the case.

3 Results

Our experiments showed that adversarial attacks can successfully deceive AI music classifiers. We evaluated the attacks based on success rate and the average number of iterations required for successful deception. As the confidence threshold increased, the success rate decreased, and the average number of iterations increased, as expected.

Our experiments showed that under unconstrained conditions, a learning rate of 1×10^{-4} achieved a success rate of 97.87% in fooling the classifier at a 50% confidence threshold, with an average of 12.67 iterations. Notably, even under constrained conditions with a minimum signal-to-noise ratio (SNR) of 50 dB, the attack maintained a success rate of 97.87%, indicating that the model could be reliably deceived without compromising perceptual quality. This high success rate under varying SNR levels underscores the vulnerability of current classifiers when exposed to targeted adversarial noise.

Additionally, our findings reveal that increasing the required minimum confidence level for misclassification (e.g., from 50% to 90%) resulted in a proportional increase in iterations, indicating the classifier's heightened resistance at higher confidence thresholds. For instance, at an SNR of 60 dB and a 90% confidence threshold, the success rate dropped to 72.34%, with an average of 20.41 iterations required. This suggests that while adversarial attacks are highly effective, increasing the classifier's confidence requirements may serve as a partial defense mechanism, albeit with diminishing returns at lower SNRs.

Furthermore, our evaluation of different hyperparameter configurations reveals that the MLP model achieved a cross-validation accuracy of 0.9714 ± 0.0128, demonstrating robust performance on the original, unaltered data. However, in the face of adversarial noise, the classifier's reliability decreased substantially. This indicates that while the classifier was well-optimized for clean data, it remains vulnerable to adversarial perturbations. This discrepancy highlights the need for adversarial training and noise-resilient embeddings in future models to maintain classification integrity.

The analysis of success rates across varied learning rates and SNR levels provides valuable insights into the dynamic interaction between attack parameters and classifier performance. A higher learning rate of 1×10^{-4} consistently led to faster convergence and higher attack success, while smaller learning rates, such as 1×10^{-6}, achieved lower success rates and required more iterations. This reinforces the importance of tuning attack parameters to maximize adversarial efficacy while balancing computational efficiency.

We also notice that the Adversarial attack sometimes fail to attack audio-lengths that are very short. This would mean that audio-length could play an important factor learned by our classifier. Moreover, we also observe that due to constraints on just the amount of noise added, the model doesn't seem to utilize the frequency aspect as well. The attack learns to add noise at intervals without showing much variation in the types of noise it adds. This means we might need more rigorous constraints for our attack to utilize this dimension as well.

Overall, the experiment results demonstrate that adversarial noise attacks, when finely tuned, can reliably exploit weaknesses in music classifiers. This has significant implications for the integrity of AI-driven music applications, such as content moderation and copyright enforcement, where such vulnerabilities could be exploited to bypass detection or enforcement mechanisms.

4 Discussion and Conclusion

This research provides a comprehensive analysis of adversarial vulnerabilities in AI-based music classifiers, specifically by leveraging CLAP embeddings and an MLP classifier to differentiate between AI-generated and human-composed music. Through extensive experiments, we demonstrated that adversarial noise can significantly undermine the classifier's performance, achieving high rates of misclassification while maintaining imperceptible noise levels for human listeners.

We notice that the Adversarial attack sometimes fail to attack audio-lengths that are very short. This would mean that audio-length could play an important factor learned by our classifier. Moreover, we also observe that due to constraints on just the amount of noise added, the model doesn't seem to utilize the frequency aspect as well. The attack learns to add noise at intervals without showing much variation in the types of noise it adds. This means we might need more rigorous constraints for our attack to utilize this dimension as well.

In conclusion, our study demonstrates that adversarial noise can effectively disguise AI-generated music, thereby revealing significant weaknesses in current AI-based music classification systems. By leveraging CLAP embeddings and a refined adversarial noise generation approach, we showed that a targeted adversarial attack could successfully mislead classifiers while remaining imperceptible to human listeners. This vulnerability, if left unaddressed, could compromise the efficacy of AI-driven applications in music, potentially leading to unintended biases or inaccuracies in music distribution, copyright enforcement, and recommendation systems.

Future work should focus on developing classifiers with built-in resilience to adversarial noise, possibly through hybrid models that integrate robustness against adversarial attacks. Additionally, expanding adversarial testing to other domains within audio classification could contribute to more secure, fair, and effective AI systems across a variety of audio-based applications.

References

[1] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R. (2014). *Intriguing properties of neural networks.* arXiv preprint arXiv:1312.6199.

[2] Subramanian, V. (2019). *Adversarial Attacks in Sound Event Classification.* arXiv preprint arXiv:1907.02477.

[3] Duan, R., Qu, Z., Zhao, S., Ding, L., Liu, Y., Lu, Z. (2022). *Perception-aware attack: Creating adversarial music via reverse-engineering human perception.* In Proceedings of the 2022 ACM SIGSAC conference on computer and communications security (pp. 905-919). Association for Computing Machinery. `https://doi.org/10.1145/3548606.3559350`.

[4] Saadatpanah, P., Shafahi, A., Goldstein, T. (2020). *Adversarial attacks on copyright detection systems.* In Proceedings of the 37th International Conference on Machine Learning (pp. 8507-8516). `https://proceedings.mlr.press/v119/saadatpanah20a.html`.

[5] Saijo, K., Ebbers, J., Germain, F. G., Khurana, S., Wichern, G., Roux, J. L. (2024). *Leveraging Audio-Only Data for Text-Queried Target Sound Extraction.* arXiv preprint arXiv:2409.13152.

[6] Elizalde, B., Deshmukh, S., Al Ismail, M., Wang, H. (2022). *CLAP: Learning audio concepts from natural language supervision.* In 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 1-5). IEEE. `https://doi.org/10.48550/arXiv.2206.04769`.