Detecting AI-Generated Audio Using Deep Learning

Trieu Vuong Nguyen
AI Development with TensorFlow
FPT University
Ho Chi Minh City, Vietnam
`vuongntse182185@fpt.edu.vn`

Abstract

The development of generative models such as VALL-E, Tacotron 2, and ElevenLabs has enabled the creation of highly realistic synthetic speech. While these technologies offer convenience and creativity in applications such as text-to-speech and virtual assistants, they also raise significant ethical concerns. AI-generated audio can be used for malicious purposes, including misinformation campaigns, impersonation fraud, and voice-based scams. This project proposes to build an intelligent audio classification system capable of distinguishing between real human speech and AI-generated audio. By leveraging deep learning techniques and training on a diverse dataset of both authentic and synthetic speech, this project aims to develop a robust and accurate detection model. The outcome of this work will contribute to the growing field of audio forensics and enhance online safety by offering a potential tool for media verification.

1 Introduction

This project focuses on developing an AI-powered system capable of detecting AI-generated audio, such as deepfake voices or synthetic speech created by text-to-speech models. The primary motivation for this project is the growing challenge posed by increasingly realistic synthetic audio, which can lead to misinformation or identity fraud. As of the milestone deadline, we've successfully completed data collection, gathering diverse examples of both human and AI-generated audio. Additionally, we've implemented a basic classification model and begun preliminary evaluations on our dataset, aiming to assess and refine the model's accuracy in distinguishing authentic human voices from synthetic ones.

2 Related Works

Recent advancements in audio classification and speech processing have prominently utilized deep learning techniques, particularly focusing on feature extraction from mel-spectrograms. MobileNetV3, a lightweight CNN model, has been widely used as a backbone for efficient feature extraction in audio tasks, balancing computational efficiency and accuracy.

In terms of model architectures, hybrid CNN-RNN models have shown promise, particularly when enhanced with self-attention mechanisms. This combination allows for efficient feature extraction and improved temporal sequence modeling, as seen in models like MobileSeqDetect. On the other hand, CNN-based models, such as SpectraNet, focus purely on convolutional layers for extracting spatial features from mel-spectrograms, offering a simpler but effective approach.

Additionally, recent works have fine-tuned pretrained models such as Wav2Vec2 and Audio Spectrogram Transformer (AST) for audio tasks. These models, known for their strong performance in capturing complex audio features and long-range dependencies, are used to enhance the capabilities of our models by further refining their ability to process and classify audio data.

3 Data Collection

For this project, I am using the The Fake-or-Real (FoR) Dataset (deepfake audio) from Kaggle, which comprises a large collection of more than 195,000 utterances. These utterances consist of both real human speech and computer-generated speech, making the dataset well-suited for binary classification tasks related to deepfake audio detection.

The dataset is available in four different versions: for-original, for-norm, for-2sec, and for-rerec. Each version serves a specific purpose depending on the desired preprocessing or use case. For this project, I utilize the for-original version, which contains the raw audio files as they were collected directly from various speech sources, without any additional normalization, trimming, or rerecording. This version is also balanced, ensuring an equal representation of real and fake samples across the dataset. The audio files are stored in .wav format and are organized into separate directories labeled as real and fake, with further subdirectories for training, validation, and testing data.

4 Method

Data Preprocessing

The preprocessing pipeline begins by filtering broken audio files using the `filter_broken_files()` function. This function attempts to load each audio file with `librosa.load()`. If a file cannot be read due to corruption, it is skipped, ensuring only valid audio files are processed.

The `get_files_labels()` function is then used to collect audio files and assign labels. It gathers files from “real” and “fake” directories, assigning a label of 1 for real audio and 0 for fake audio. The paths to these files are fetched using TensorFlow’s `tf.io.gfile.glob()`.

For each audio file, the `preprocess_audio()` function performs several transformations. The audio is first loaded, then padded or cropped to a fixed length of 48,000 samples (3 seconds at 16 kHz). It is converted into a Mel-spectrogram, and the values are transformed into a logarithmic scale (dB), with normalization applied to scale the values between 0 and 1, making it suitable for model input.

Custom wrappers, such as `train_preprocess_wrapper()` and `val_test_preprocess_wrapper()`, apply the preprocessing function to the datasets. These wrappers use `tf.py_function()` to incorporate the custom audio processing logic into TensorFlow’s data pipeline, ensuring that data is appropriately processed for training, validation, and testing.

Finally, the `create_dataset()` function batches, shuffles (for training), and prefetches the processed data into TensorFlow datasets. It ensures efficient data handling and feeding into the model during training by using parallel processing for data loading.

In order to prepare the Mel-spectrogram for training, the `preprocess_mel_spectrogram()` function resizes it to 224x224 pixels and replicates it across three channels, making it suitable for models that require 3-channel input. This ensures the Mel-spectrogram is in the proper format for deep learning.

SpectraNet

The `SpectraNet` model is designed for audio analysis, specifically for detecting AI-generated audio using Mel-spectrograms as input.

- **Backbone:** The model uses a pre-trained `MobileNetV3Large` model (without the top classification layer) for feature extraction. The backbone is frozen, meaning only the custom layers will be trained.
- **Input Layer:** The input is a Mel-spectrogram of shape $(128, 94, 1)$, which is passed through a `Lambda` layer to preprocess the spectrogram by resizing and replicating it to fit the backbone input size $(224, 224, 3)$.

- **Feature Extraction:** The processed spectrogram is passed through the MobileNetV3 backbone, followed by a `GlobalAveragePooling2D` layer to reduce the spatial dimensions.
- **Fully Connected Layers:** Two dense layers with 256 units, ReLU activation, L2 regularization, BatchNormalization, and Dropout (0.3) are applied to the pooled features.
- **Output Layer:** A final dense layer with a sigmoid activation function outputs a value between 0 and 1 for binary classification (real vs. fake audio).

This model leverages the efficiency of MobileNetV3, with added regularization techniques to avoid overfitting, making it suitable for audio classification tasks.

MobileSeqDetect

The `MobileSeqDetect` model is designed for audio analysis using Mel-spectrograms for detecting AI-generated audio.

- **Input Layer:** The model takes a Mel-spectrogram of shape $(128, 94, 1)$, which is processed using a `Lambda` layer to resize and replicate it to fit the MobileNetV3 backbone input size $(224, 224, 3)$.
- **Backbone:** A pre-trained `MobileNetV3Large` model (with weights from ImageNet) is used as the feature extractor. The backbone is frozen during training.
- **Sequence Modeling:** The output of the backbone is reshaped and passed through a `Bidirectional LSTM` layer with 256 units, followed by an attention mechanism for improved feature extraction.
- **Pooling and Dense Layers:** The attention output is pooled using `GlobalAveragePooling1D`, and two dense layers with 256 units, ReLU activation, L2 regularization, BatchNormalization, and Dropout (0.3) are applied.
- **Output Layer:** A final dense layer with a sigmoid activation function outputs a value between 0 and 1, performing binary classification (real vs. fake audio).

This model combines the power of MobileNetV3 for feature extraction with sequence modeling using LSTMs and attention mechanisms to effectively handle audio data for classification.

Wav2Vec2

The `Wav2Vec` model used in this task is based on the pre-trained `facebook/wav2vec2-base` model, fine-tuned for audio classification. It is specifically designed to process raw audio data and classify it into predefined labels.

- **Model Architecture:** The model is loaded using the `AutoModelForAudioClassification` class from Hugging Face's `transformers` library. It is fine-tuned for a custom task with a specific number of output labels. The label mappings are provided via `label2id` and `id2label` dictionaries, ensuring correct mapping between model outputs and class labels.
- **Training Arguments:** The training setup is defined using the `TrainingArguments` class, which specifies various hyperparameters:
  - `eval_strategy` and `save_strategy` are set to `epoch`, meaning evaluations and model checkpoints are done at the end of each training epoch.
  - `learning_rate` is set to 3×10^{-5}, a common choice for fine-tuning pre-trained models.
  - `per_device_train_batch_size` and `per_device_eval_batch_size` are both set to 32, with gradient accumulation over 4 steps to simulate a larger batch size.
  - `num_train_epochs` is set to 5, and `warmup_ratio` is set to 0.1 to help stabilize training in the initial steps.
  - `logging_steps` defines how frequently logs are printed, set to every 10 steps.
  - The best model is automatically loaded at the end of training, and `metric_for_best_model` is set to `accuracy`, ensuring the model with the highest accuracy is selected.

- **Metric Computation:** The model uses accuracy as the evaluation metric. The `compute_metrics` function computes accuracy based on predictions made by the model. The predictions are obtained by applying `np.argmax` to the model's output logits to select the most likely class for each sample.
- **Trainer:** The training and evaluation process is handled by the `Trainer` class from the `transformers` library. The trainer is initialized with the model, training arguments, training and validation datasets, tokenizer (feature extractor), and the metrics function.

This setup allows for efficient fine-tuning of the `Wav2Vec` model for the audio classification task, leveraging pre-trained representations and fine-tuning them for the specific dataset at hand.

Audio Spectrogram Transformer

The `ASTForAudioClassification` model leverages the Audio Spectrogram Transformer (AST) architecture, fine-tuned for audio classification tasks. This model uses the pre-trained `MIT/ast-finetuned-audioset-10-10-0.4593` model, which has been specifically trained on the AudioSet dataset for classification tasks.

- **Model Architecture:** The model is loaded using the `ASTForAudioClassification` class from Hugging Face's `transformers` library. The weights from the pre-trained `MIT/ast-finetuned-audioset-10-10-0.4593` model are fine-tuned for the custom audio classification task, with the number of output labels defined by `num_labels`. The mappings between labels and their corresponding IDs are provided by the `label2id` and `id2label` dictionaries.
- **Training Arguments:** The training configuration is set using the `TrainingArguments` class:
  - `eval_strategy` and `save_strategy` are set to `epoch`, so the model is evaluated and saved at the end of each epoch.
  - The learning rate is 3×10^{-5}, with a batch size of 8. Gradient accumulation over 4 steps is used to simulate larger batch sizes.
  - `num_train_epochs` is set to 3, and `warmup_ratio` is set to 0.1 to stabilize training early.
  - The model is evaluated using accuracy, and the best model is loaded based on this metric using `load_best_model_at_end = True`.
- **Metric Computation:** The evaluation metric used is accuracy, computed by applying `np.argmax` to the model's predictions to select the class with the highest probability.
- **Trainer:** The model is trained and evaluated using Hugging Face's `Trainer` class. This class manages the training loop, logging, model evaluation, and the application of the compute metrics function.

This configuration enables fine-tuning the pre-trained AST model, which excels at audio classification tasks, by training it on custom datasets with efficient evaluation and logging strategies.

5 Results and Findings

Training Results

The following graphs show the training and validation accuracy and loss curves for the models `MobileSeqDetect`, `SpectraNet`, `AST`, and `Wav2Vec2` across several epochs.

- **MobileSeqDetect:**
  - **Accuracy:** The training accuracy remains constant at around 0.5, with validation accuracy fluctuating significantly.
  - **Loss:** Training loss is low, while validation loss fluctuates, indicating potential overfitting or unstable learning.
- **SpectraNet:**

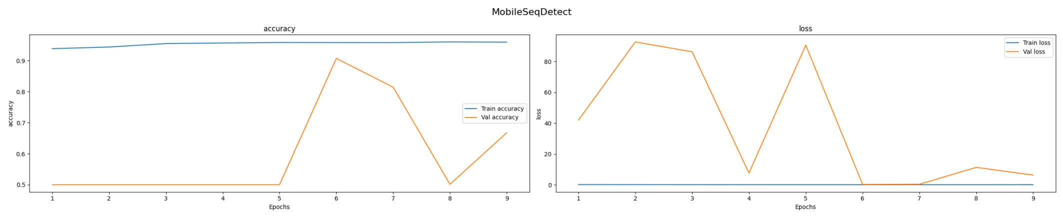

Figure 1: Training and Validation Accuracy and Loss for MobileSeqDetect

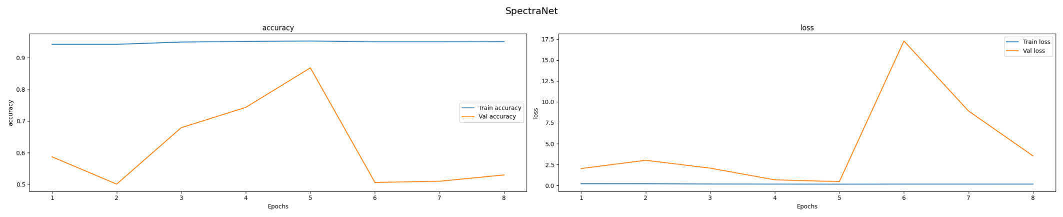

Figure 2: Training and Validation Accuracy and Loss for SpectraNet

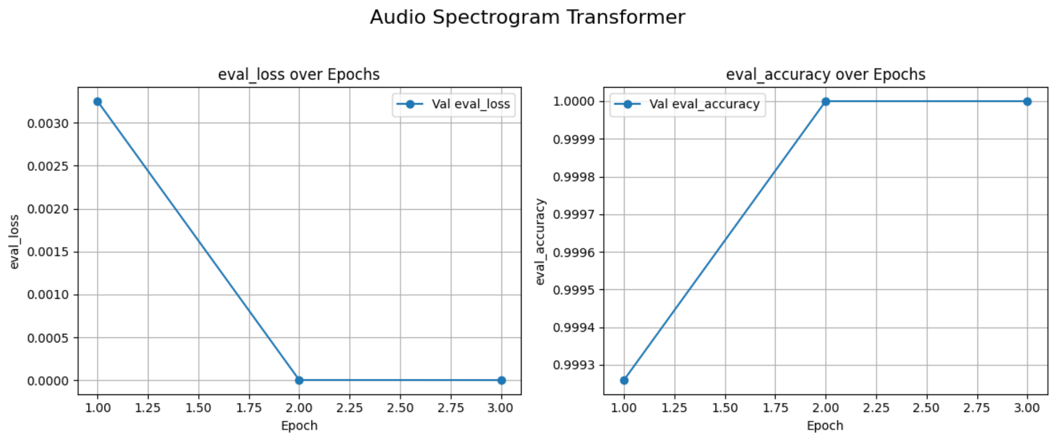

Figure 3: Training and Validation Accuracy and Loss for Audio Spectrogram Transformer (AST)

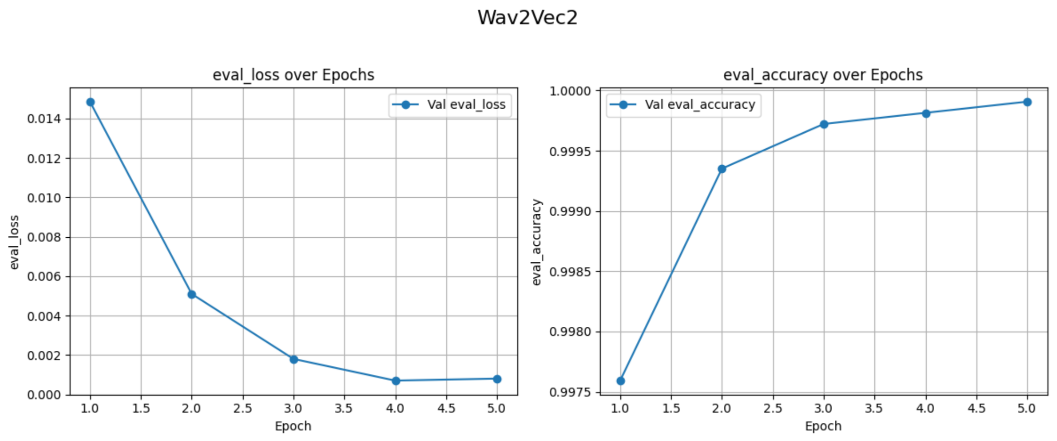

Figure 4: Training and Validation Accuracy and Loss for Wav2Vec2

– **Accuracy:** The training accuracy remains high and stable, while the validation accuracy fluctuates but shows some improvement.

  - **Loss:** Training loss is nearly constant, while validation loss fluctuates in the early epochs.
- **AST:**
  - **Accuracy:** The validation accuracy increases steadily after the first epoch, reaching near 1.0 by epoch 3.
  - **Loss:** The validation loss drops significantly, reaching near 0 by epoch 2.
- **Wav2Vec2:**
  - **Accuracy:** The validation accuracy improves steadily across the 5 epochs, approaching 1.0.
  - **Loss:** The validation loss decreases steadily, reaching near 0 by the 5th epoch.

Evaluation

The following table summarizes the performance metrics for the models `SpectraNet`, `MobileSeqDetect`, `Wav2Vec2`, and `AST`.

Model	Accuracy	Precision	F1 Score	Recall	ROC AUC
`SpectraNet`	0.6454	0.7478	0.5999	0.6454	0.6378
`MobileSeqDetect`	0.7190	0.7201	0.7182	0.7190	0.7179
`Wav2Vec2`	0.5311	0.7607	0.4086	0.5311	0.5416
`AST`	0.4892	0.7503	0.3221	0.4892	0.5006

Table 1: Comparison of Evaluation Metrics for Different Models

Below are the confusion matrices for each model, providing further insight into their classification performance.

6 Discussion

Based on the evaluation metrics and confusion matrices for `SpectraNet`, `MobileSeqDetect`, `Wav2Vec2`, and `AST`, we can summarize the performance of these models as follows:

- **SpectraNet:**
  - **Accuracy:** 64.5%, indicating moderate performance.
  - **Precision:** 74.8%, suggesting that when SpectraNet classifies an instance as positive, it is correct most of the time.
  - **Recall:** 64.5%, which is equal to accuracy in this case, showing balanced classification performance.
  - **F1 Score:** 59.99%, reflecting the trade-off between precision and recall.
  - **ROC AUC:** 63.8%, indicating a fair ability to distinguish between classes.

  Despite its reasonable precision and recall, SpectraNet struggles with classification accuracy and does not generalize well, as shown by the fluctuations in the confusion matrix.
- **MobileSeqDetect:**
  - **Accuracy:** 71.9%, showing a higher classification accuracy than SpectraNet.
  - **Precision:** 72.0%, implying it correctly identifies positive cases effectively.
  - **Recall:** 71.9%, aligning with accuracy, showing good overall recall.
  - **F1 Score:** 71.8%, demonstrating balanced performance between precision and recall.
  - **ROC AUC:** 71.8%, indicating strong discriminatory ability.

  MobileSeqDetect demonstrates better consistency in terms of overall accuracy and a more balanced classification compared to SpectraNet, making it a stronger choice for the task.
- **Wav2Vec2:**
  - **Accuracy:** 53.1%, lower than both SpectraNet and MobileSeqDetect, suggesting room for improvement.

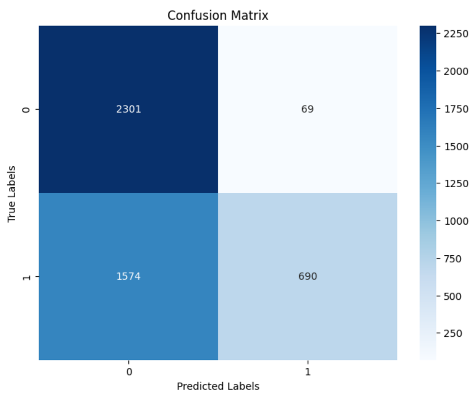

Figure 5: Confusion Matrix for SpectraNet

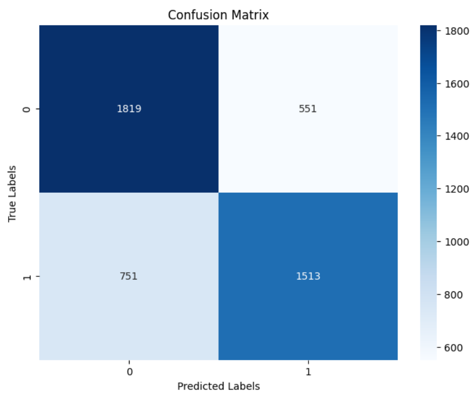

Figure 6: Confusion Matrix for MobileSeqDetect

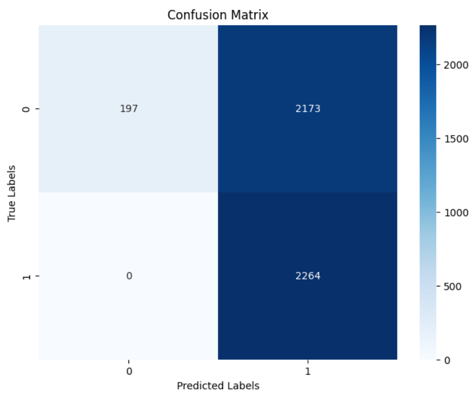

Figure 7: Confusion Matrix for Wav2Vec2

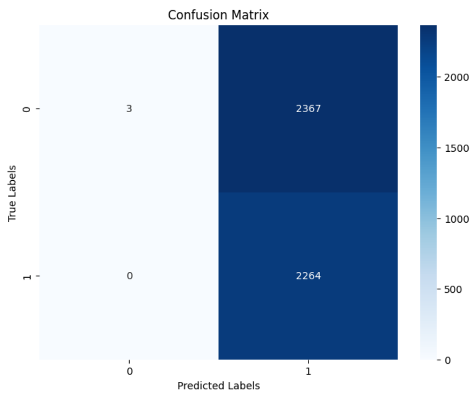

Figure 8: Confusion Matrix for AST

  - **Precision:** 76.1%, high precision means Wav2Vec2 is accurate when it classifies positive instances, but...
  - **Recall:** 53.1%, the recall is low, indicating that Wav2Vec2 misses many positive instances.
  - **F1 Score:** 40.86%, indicating poor trade-off between precision and recall.
  - **ROC AUC:** 54.16%, reflecting a suboptimal ability to distinguish between classes.

  Wav2Vec2 struggles in classification performance, particularly with recall and F1 score, despite achieving good precision. It may need further tuning or model adjustments to improve its recall and overall performance.

- **AST:**
  - **Accuracy:** 48.9%, the lowest among the models, suggesting significant issues with classification.
  - **Precision:** 75.0%, indicating high precision when it classifies instances as positive, but...
  - **Recall:** 48.9%, on par with accuracy, showing that AST fails to capture many positive instances.
  - **F1 Score:** 32.21%, a poor trade-off between precision and recall.
  - **ROC AUC:** 50.06%, barely above random chance.

  AST shows the lowest performance across all metrics, particularly with F1 score and recall. It seems to struggle in the classification task, likely due to poor generalization.

Final Thoughts

- **MobileSeqDetect** stands out as the best model overall, with the highest accuracy, precision, recall, F1 score, and ROC AUC.
- **SpectraNet** performs reasonably well but lags behind MobileSeqDetect.
- **Wav2Vec2** and **AST** underperform, particularly in terms of recall and F1 score, with AST being the weakest model in this comparison.

Improving the performance of **Wav2Vec2** and **AST** could involve further model tuning, regularization, or possibly training with a larger or more diverse dataset. For tasks requiring high accuracy and balanced performance, **MobileSeqDetect** is the most reliable model based on this evaluation.

Errors and Limitations

- **Overfitting:** Some models, particularly `SpectraNet`, show potential overfitting, as indicated by the fluctuating validation accuracy and loss. This can be addressed through:
  - Early stopping.
  - Data augmentation or regularization techniques like dropout and weight decay.
- **Model Capacity:** `Wav2Vec2` and `AST` might be struggling due to their architecture or insufficient fine-tuning for this specific task.
- **Data Quality and Size:** The performance of `Wav2Vec2` and `AST` may suffer from insufficient or imbalanced training data. Increasing the dataset size and improving data labeling could lead to better performance.
- **Evaluation Metrics:** Some models exhibit high `precision` but poor `recall`, indicating that they are conservative in predicting the positive class. Future work could explore different evaluation metrics like `Precision-Recall AUC`.

Future Improvements

- **Improve Wav2Vec2 and AST Performance:**
  - Fine-tune these models on a larger, more diverse dataset.
  - Adjust the model architectures to better fit the task (e.g., adding more layers, attention mechanisms).
- **Fine-tune Hyperparameters:** Both `SpectraNet` and `Wav2Vec2` would benefit from hyperparameter optimization, such as adjusting the learning rate, batch size, or the number of epochs for training.
- **Regularization and Early Stopping:** Introduce early stopping and experiment with different regularization strategies like L2 regularization or adding dropout layers.

Final Summary

In this work, we evaluated and compared the performance of four audio classification models: `SpectraNet`, `MobileSeqDetect`, `Wav2Vec2`, and `AST`. The evaluation was based on key metrics such as accuracy, precision, recall, F1 score, and ROC AUC. Each model was tested on a dataset of audio spectrograms to classify real and fake audio signals.

Key Findings:

- `MobileSeqDetect` outperformed the other models, with the highest accuracy (71.9%) and balanced precision and recall. It showed strong generalization and high discriminatory power (ROC AUC = 71.8%).
- `SpectraNet` performed reasonably well, with good precision (74.8%) but struggled with recall and generalization, as seen from fluctuating validation loss.
- `Wav2Vec2` and `AST` underperformed significantly, particularly in terms of recall and F1 score. `Wav2Vec2` had high precision but missed many positive cases, while `AST` showed poor performance across all metrics.

Limitations:

- Overfitting: Some models, especially `SpectraNet`, showed signs of overfitting due to fluctuating validation performance.
- Model Capacity: `Wav2Vec2` and `AST` may require further fine-tuning or architectural adjustments to perform well on the given task.
- Data Quality: The quality and size of the training dataset could have impacted the performance of `Wav2Vec2` and `AST`.

Future Improvements:

- Further fine-tuning and data augmentation for `Wav2Vec2` and `AST`.
- Regularization techniques like dropout, weight decay, and early stopping to prevent overfitting.
- Hyperparameter optimization to improve model stability and performance.

In conclusion, `MobileSeqDetect` stands as the most effective model for this classification task. With further improvements in `Wav2Vec2` and `AST`, their performance could be enhanced, making them competitive models in audio classification tasks.

References

[1] Howard, A., Sandler, M., Chu, G., Chen, B., Tan, M., Wang, W., Zhu, Y. (2019). *Searching for MobileNetV3*. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 1314-1324.

[2] Baevski, A., Zhou, H., Mohamed, A., Auli, M. (2020). *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations*. In Proceedings of NeurIPS 2020.

[3] Gao, Y., Zhang, Y., Song, L. (2021). *AST: Audio Spectrogram Transformer*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 1638-1647.