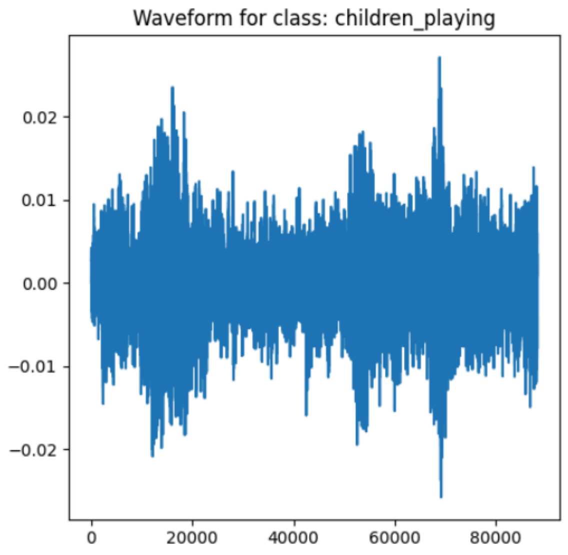
REPORT
Assignment - Sound Classification

LITERATURE:

Sound classification is a task in the field of audio signal processing and machine learning where the goal is to categorize audio signals into different classes or categories based on their content. This can be useful in various applications, such as speech recognition, music genre classification, environmental sound analysis, and more.

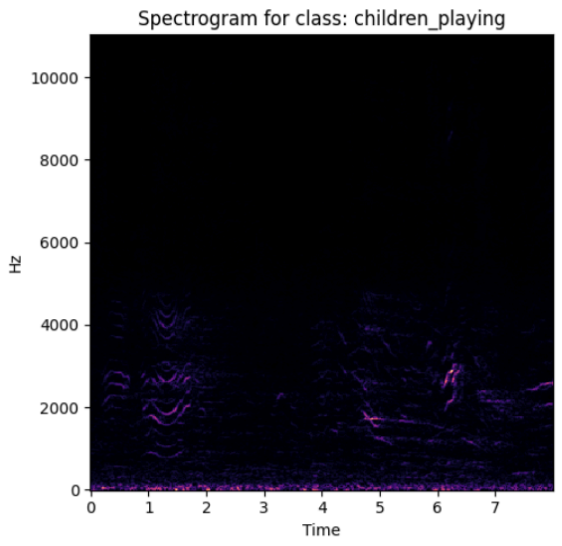
Sound is the term to describe what is heard when sound waves pass through a medium to the ear. All sounds are made by vibrations of molecules through which the sound travels.

The amplitude of the vibrations defines the loudness of the sound and frequency is the speed with which the amplitude changes.



A **spectrogram** is a visual representation of the spectrum of frequencies of a signal as it varies with time. The spectrogram is a concise ‘snapshot’ of an audio wave and since it is an image, it is well suited to being input to CNN-based architectures developed for handling images.

It plots Frequency (y-axis) vs Time (x-axis) and uses different colors to indicate the Amplitude of each frequency. The brighter the color the higher the energy of the signal.

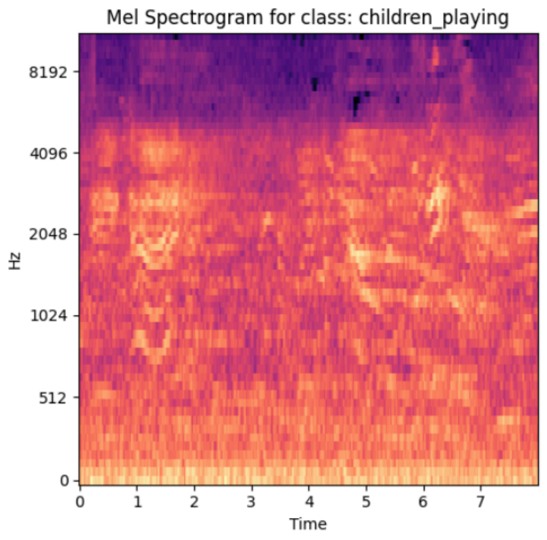


The **Mel Scale** was developed to take this into account by conducting experiments with a large number of listeners. It is a scale of pitches, such that each unit is judged by listeners to be equal in pitch distance from the next.

The human perception of the amplitude of a sound is its loudness. And similar to frequency, we hear loudness logarithmically rather than linearly. We account for this using the **Decibel scale**.

A **Mel Spectrogram** makes two important changes relative to a regular spectrogram that plots Frequency vs Time.

- It uses the Mel Scale instead of Frequency on the y-axis.
- It uses the Decibel Scale instead of Amplitude to indicate colors.



Training deep learning models requires a lot of computational power and time, and re-training such models from scratch to include the newly obtained data is often difficult. This is where the concept of “Zero-Shot Learning” comes to the rescue.

Zero-shot learning is a machine learning paradigm where a model is trained to recognize and generalize to classes that it has never seen during training. In traditional supervised learning, a model is trained on a labeled dataset that includes examples from all classes it is expected to recognize. However, in zero-shot learning, the model is designed to handle cases where new classes, not seen during training, need to be recognized and classified.

The key idea behind zero-shot learning is to leverage additional information or side information, often referred to as "auxiliary information" or "semantic information," about both the seen and unseen classes. This auxiliary information might include textual descriptions, attributes, word embeddings, or any other form of information that characterizes the classes.

Fine-tuning refers to the process of taking a pre-trained machine learning model and further training it on a new, specific task or dataset. The goal of fine-tuning is to leverage the knowledge gained by a model on a large and general dataset (pre-training) and adapt it to perform well on a more specific task or dataset.

- The pre-trained model is further trained on a task-specific dataset, called the target domain or fine-tuning dataset.
- The weights of the pre-trained model are adjusted based on the task-specific data, allowing the model to adapt to the nuances and characteristics of the new task.
- The goal is to refine the model's parameters to improve its performance on the target task while preserving the knowledge gained during pre-training.

Fine-tuning is particularly useful in scenarios where labeled data for the target task is limited, as the model can leverage the broader knowledge acquired during pre-training.

<u>APPROACHES STUDIED:</u>

- For task 1 i.e., zero-shot model, openL3 model has been used.

  OpenL3 is an open-source deep audio embedding model designed for various audio analysis tasks. It provides fixed-size embeddings for audio signals, which can be used for tasks like audio classification, similarity matching, and retrieval. OpenL3 uses a deep convolutional neural network (CNN) architecture to generate these embeddings.

  In this approach we first have found the audio embeddings for the audios and then using the linear classifier model(SVM) have evaluated and found the metrics.

- For task 2 i.e., training AM and LM, resnet50 and whisper models have been used.

  ResNet50:

  In this wave are converted into spectrograms and then normalized to form an image. ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pre-trained version of the neural network trained on more than a million images from the ImageNet database. The pre-trained neural network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. The neural network has an image input size of 224-by-224.

  In this approach of fine tuning we have changed the first layer so that the input can be passed to the model and the last layer for classification is added with 10 as the output classes. After this the model is trained on the dataset and evaluated on the test set.

  Whisper:

  Whisper is an automatic speech recognition (ASR) system trained on 6,80,000 hours of multilingual and multitask supervised data collected from the web. It enables transcription in multiple languages, as well as translation from those languages into English.

  The models were trained on either English-only data or multilingual data. The English-only models were trained on the task of speech recognition. The multilingual models were trained on both speech recognition and speech translation. For speech recognition, the model predicts transcriptions in the same language as the audio. For speech translation, the model predicts transcriptions to a different language to the audio.

In this approach the above Whisper model architecture a classifier is built for the audio classification with the last layer giving the output for the number of classes. After this the model is trained on the dataset and evaluated on the test set.

HYPOTHESIS:

1. Different sound classes exhibit distinct frequency patterns: Audio events or categories may be characterized by specific frequency components. For example, certain instruments in music might occupy specific frequency ranges.

2. The shape of the audio waveform is correlated with specific sound classes: Certain sound events might be characterized by unique waveform shapes. For example, abrupt changes in a waveform might be indicative of percussive sounds.

3. Temporal variations in spectrograms capture dynamic changes over time that are relevant for sound classification. Sound events often have characteristic temporal patterns, and these temporal dynamics should be reflected in the spectrogram.

4. Mel spectrograms effectively localize and highlight important frequency regions relevant to sound classification: Mel spectrograms emphasize certain frequency bands while de-emphasizing others, potentially highlighting characteristics crucial for discriminating between different sound classes.

5. The contrast between different frequency regions in mel spectrograms provides discriminative information for sound classification:Regions of high contrast in the spectrogram might be indicative of important features or patterns in the audio signal.

FINDING AND RESULTS:

Zero Shot Learning:
(Model - OpenL3)

Advantages:

1. Generalization to Unseen Classes: Zero-shot learning allows the model, trained on a set of seen classes, to generalize and make predictions on completely unseen classes without any explicit training data for those classes. This is particularly useful in scenarios where new classes may emerge or be added over time.

2. Utilization of OpenL3 Features: OpenL3 extracts deep audio embeddings that capture rich and discriminative information from audio signals. These embeddings can be used as semantic representations for zero-shot learning, providing a basis for recognizing unseen classes.

Disadvantages:

1. Limited Fine-Tuning for Specific Characteristics: OpenL3 features might not be fine-tuned specifically for the characteristics of the target domain or unseen classes. This could lead to suboptimal performance compared to a model fine-tuned on task-specific data.

2. Dependency on Embedding Quality: The success of zero-shot learning relies on the quality of the semantic embeddings provided by OpenL3. If these embeddings fail to capture relevant information for certain classes, the model's performance may be limited.

Fine Tuning:
(Model - Resnet50)

Advantages:

1. Transfer of Pre-Trained Visual Features: ResNet50, originally designed for visual tasks, captures hierarchical features that might also be relevant for audio classification. Fine-tuning allows the model to transfer these pre-trained visual features to the audio domain.

2. Transfer Learning Efficiency: Fine-tuning is often more computationally efficient than training a model from scratch, especially when working with limited labeled audio data. The pre-trained ResNet50 model provides a valuable starting point, reducing the need for a large amount of task-specific data.

Disadvantages:

1. Domain Shift Challenges: If there is a significant domain shift between the pre-training data of ResNet50 (usually visual data) and the target audio domain, fine-tuning may face challenges. The model might not generalize well to audio-specific characteristics.

2. Potential for Negative Transfer: Fine-tuning may lead to negative transfer if the pre-trained visual features are not relevant to the audio classification task. This could result in a model that performs poorly compared to training a task-specific model.

Fine Tuning:
(Model - Whisper)

Advantages:

1. Task-Specific Adaptation: Fine-tuning allows the model to adapt to the specific characteristics of the target audio classification task. The Whisper model, being pre-trained on a diverse dataset, can benefit from further training on task-specific data.

2. Utilization of Pre-Trained Features: The Whisper model extracts high-level features from audio signals during pre-training. Fine-tuning leverages these learned features, providing a solid foundation for capturing relevant information in the target domain.

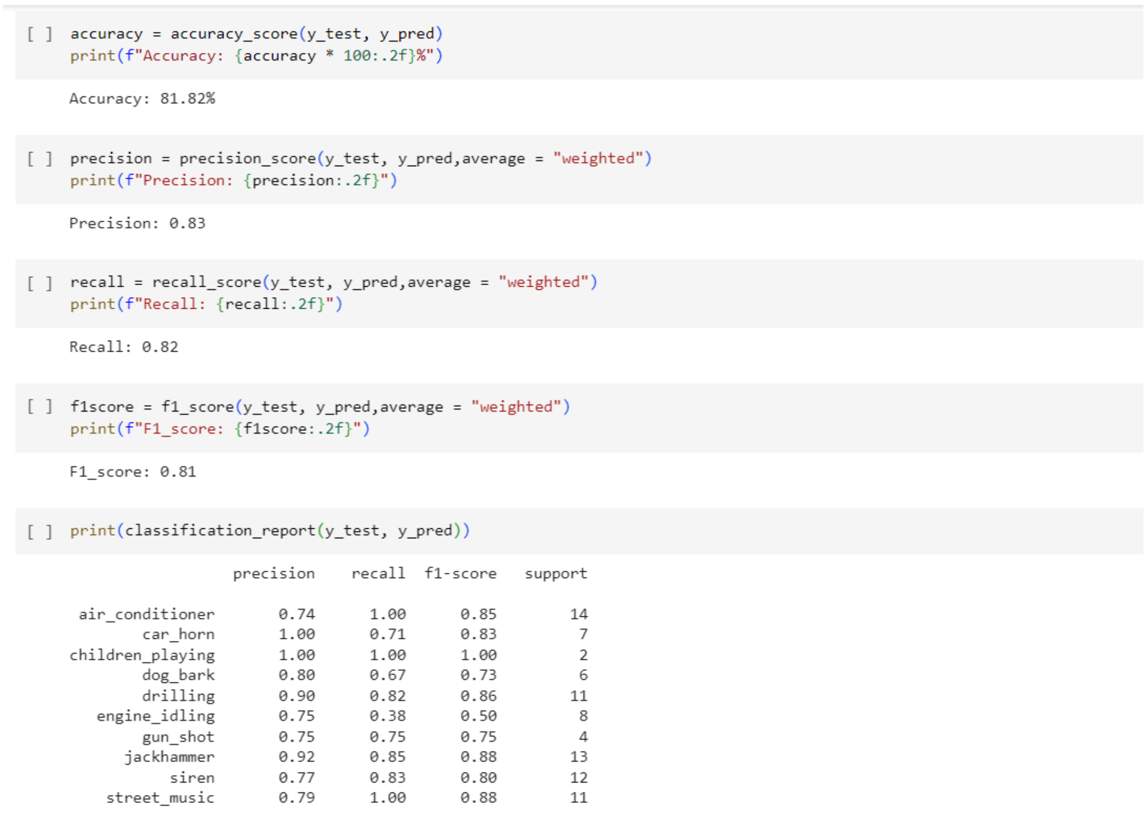
Disadvantages:

1. Computationally Intensive: Fine-tuning can be computationally intensive, especially when dealing with large models like Whisper. This may be a limitation in scenarios where computational resources are constrained.

2. Overfitting to Task-Specific Data: Fine-tuning may lead to overfitting, especially when the task-specific dataset is small. The model might memorize noise or specific patterns from the training data that do not generalize well to unseen examples.

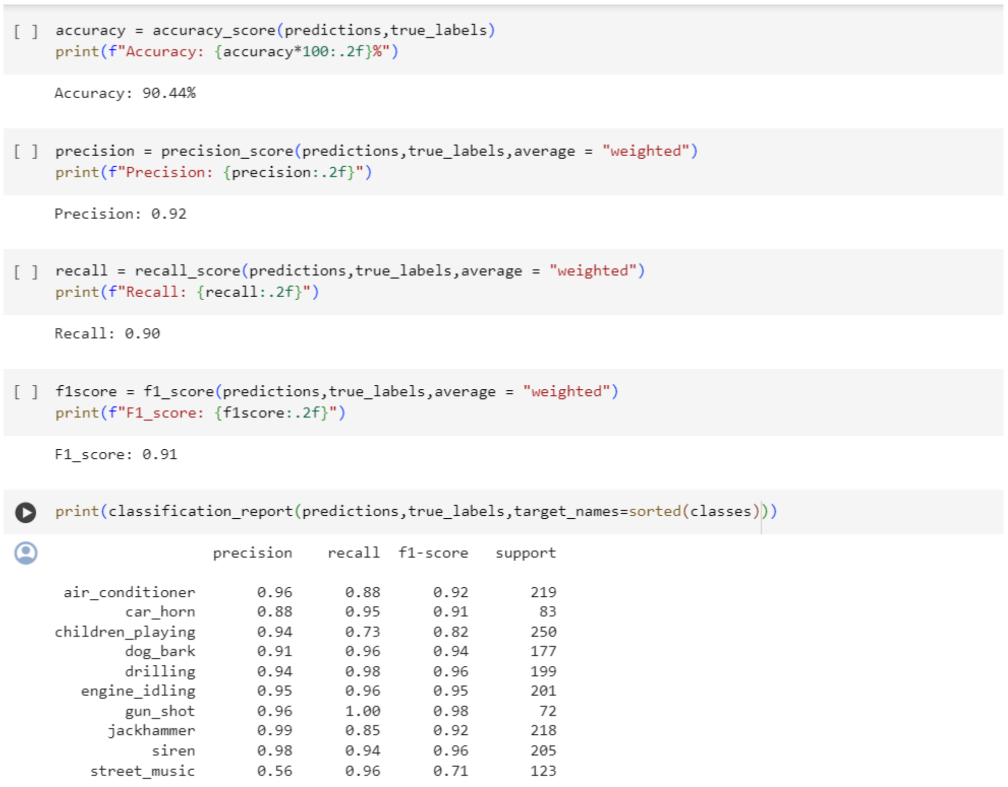
Results:

Zero Shot Learning:



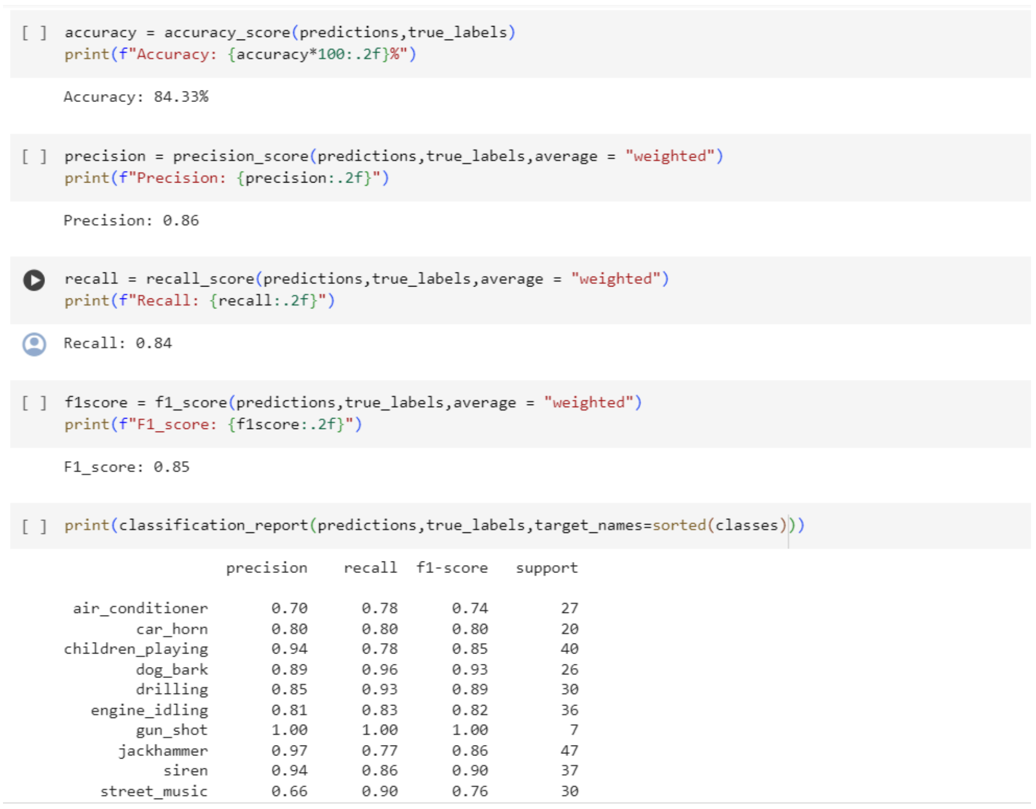
```
[ ] accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy * 100:.2f}%")

    Accuracy: 81.82%

[ ] precision = precision_score(y_test, y_pred,average = "weighted")
    print(f"Precision: {precision:.2f}")

    Precision: 0.83

[ ] recall = recall_score(y_test, y_pred,average = "weighted")
    print(f"Recall: {recall:.2f}")

    Recall: 0.82

[ ] f1score = f1_score(y_test, y_pred,average = "weighted")
    print(f"F1_score: {f1score:.2f}")

    F1_score: 0.81

[ ] print(classification_report(y_test, y_pred))

                      precision    recall  f1-score   support

     air_conditioner       0.74      1.00      0.85        14
            car_horn       1.00      0.71      0.83         7
    children_playing       1.00      1.00      1.00         2
            dog_bark       0.80      0.67      0.73         6
            drilling       0.90      0.82      0.86        11
       engine_idling       0.75      0.38      0.50         8
             gun_shot       0.75      0.75      0.75         4
          jackhammer       0.92      0.85      0.88        13
               siren       0.77      0.83      0.80        12
        street_music       0.79      1.00      0.88        11
```

Fine Tuning:
ResNet50:



```
[ ] accuracy = accuracy_score(predictions,true_labels)
    print(f"Accuracy: {accuracy*100:.2f}%")

    Accuracy: 90.44%

[ ] precision = precision_score(predictions,true_labels,average = "weighted")
    print(f"Precision: {precision:.2f}")

    Precision: 0.92

[ ] recall = recall_score(predictions,true_labels,average = "weighted")
    print(f"Recall: {recall:.2f}")

    Recall: 0.90

[ ] f1score = f1_score(predictions,true_labels,average = "weighted")
    print(f"F1_score: {f1score:.2f}")

    F1_score: 0.91

    print(classification_report(predictions,true_labels,target_names=sorted(classes)))

                      precision    recall  f1-score   support

    air_conditioner       0.96      0.88      0.92       219
           car_horn       0.88      0.95      0.91        83
   children_playing       0.94      0.73      0.82       250
           dog_bark       0.91      0.96      0.94       177
           drilling       0.94      0.98      0.96       199
      engine_idling       0.95      0.96      0.95       201
           gun_shot       0.96      1.00      0.98        72
         jackhammer       0.99      0.85      0.92       218
              siren       0.98      0.94      0.96       205
       street_music       0.56      0.96      0.71       123
```

<u>**Whisper**</u>



```
[ ] accuracy = accuracy_score(predictions,true_labels)
    print(f"Accuracy: {accuracy*100:.2f}%")

    Accuracy: 84.33%

[ ] precision = precision_score(predictions,true_labels,average = "weighted")
    print(f"Precision: {precision:.2f}")

    Precision: 0.86

    recall = recall_score(predictions,true_labels,average = "weighted")
    print(f"Recall: {recall:.2f}")

    Recall: 0.84

[ ] f1score = f1_score(predictions,true_labels,average = "weighted")
    print(f"F1_score: {f1score:.2f}")

    F1_score: 0.85

[ ] print(classification_report(predictions,true_labels,target_names=sorted(classes)))

                      precision    recall  f1-score   support

    air_conditioner       0.70      0.78      0.74        27
           car_horn       0.80      0.80      0.80        20
   children_playing       0.94      0.78      0.85        40
           dog_bark       0.89      0.96      0.93        26
           drilling       0.85      0.93      0.89        30
      engine_idling       0.81      0.83      0.82        36
           gun_shot       1.00      1.00      1.00         7
         jackhammer       0.97      0.77      0.86        47
              siren       0.94      0.86      0.90        37
       street_music       0.66      0.90      0.76        30
```

Overall the accuracy of zero shot(81.82) is slightly lower than fine tuning(90.44,84.33). This can be because fine tuning performs more effectively for this stable task and where sufficient labeled data for target class is available, but zero shot learning is done on a subset due to limits so might be a chance it could have performed better.

In zero shot the classes having precision 1 and recall 1 are more as compared to fine tuning this means for some classes zero shot generalizes better than the fine tuning task.

Here accuracy of Whisper model is lower than the ResNet50 model this can be because of limited compute the Whisper model has been fine tuned on a small subset of data and if the whole data could be provided it could have generalized much better than ResNet50 and could have yielded more accuracy.