Project Task: Feature Fusion Using Raw Signal and Spectrogram

For the project, you may use the same dataset and data preparation code that were used in Assignment 2. You have already designed and trained a 1D CNN using raw time-domain signals in Assignment 1, and a 2D CNN using spectrograms in Assignment 2. In this project, you will extend those ideas by combining both sources of information through a feature-level fusion approach.

The goal of this task is to investigate whether combining time-domain and time–frequency representations improves classification performance compared to using either representation alone.

Fusion Concept

Raw time-domain signals and spectrograms provide complementary information:

• The raw 1D signal captures waveform shape, temporal patterns, and local variations in amplitude.
• The spectrogram captures frequency content, harmonic structure, and energy distribution over time.

By fusing features extracted from both representations, the model can learn richer and more discriminative representations.

For conceptual idea, you can read this paper, but it will not help you in implementation.

Input to the Fusion Model

For each sample in the dataset, your data loader must return three elements:

1. A raw signal segment represented as a 1D tensor of shape (T).
2. A spectrogram represented as a 2D tensor of shape (1, F, T_spec).
3. A class label corresponding to the signal category.

During training, batches will therefore have the following shapes:

Raw signals: (B, T)
Spectrograms: (B, 1, F, T_spec)
Labels: (B,)

You must ensure that the raw signal and spectrogram correspond to the same original sample.

Model Architecture: Two-Stream Late Fusion

You will implement a two-stream neural network consisting of:

You will implement a two-stream architecture with:

1. RawBranch1D: a 1D CNN that outputs an embedding vector

   Output shape: (B, emb)

2. SpecBranch2D: a 2D CNN that outputs an embedding vector

   Output shape: (B, emb)

3. Fusion Head: concatenate the embeddings and classify

   Concatenation:

   - zr: (B, emb)

   - zs: (B, emb)

   - z = concat([zr, zs]) → (B, 2*emb)

Then pass z through a small architecture to predict class logits:

- output shape: (B, num_classes)

Both branches output feature vectors of the same dimension. These vectors are concatenated and passed through a fully connected network to predict the class probabilities. This approach is known as late fusion, because fusion is performed after feature extraction rather than at the input level.

Models to Train

You are required to train and evaluate the following three models:

1. RawOnlyModel: uses only the raw signal branch.

2. SpecOnlyModel: uses only the spectrogram branch.

3. TwoStreamFusionModel: uses both branches and feature fusion.

All models must be trained using the same training, validation, and test splits, as well as the same optimization settings (learning rate, batch size, number of epochs).

Evaluation and Reporting

Your project report must include:

• Performance comparison of all three models using accuracy and/or F1-score.
• Confusion matrices for each model.
• A discussion explaining whether feature fusion improves performance and why.
• Observations on which classes benefit most from fusion.