2. A Simple Classifier-Based SED System

Our starting point was a pre-trained 1D-CNN model (CNN1D\_testBACC\_0.8201.pth) from the previous project phase. This model was trained to produce frame-level predictions (120ms), which needed to be adapted for the task's 1.2-second segment-level prediction requirement.

2.1 Aggregation and Thresholding

To create a simple, unoptimized classifier baseline, we followed a two-step process:

Aggregation: We combined the 10 frame-level probability outputs within each 1.2-second segment into a single segment-level probability. For this simple system, we used a max aggregation function, taking the highest probability value across the 10 frames as the representative probability for the segment.

Thresholding: A standard, naive decision threshold of 0.5 was applied to the aggregated probabilities for all 10 classes to produce a binary prediction.

2.2 Cost Minimization Strategy

At this initial stage, no advanced cost-specific minimization strategies were applied. The goal was purely to assess the raw predictive power of the model using a standard approach before introducing targeted optimizations.

2.3 Performance vs. Naive Baseline

This simple SED system achieved a validation cost of 55.34. This result is a dramatic improvement over the naive baseline's cost of 107.78, confirming that the underlying classifier is effective and provides a strong foundation for further refinement. However, the cost is still considerably high, indicating that the naive 0.5 threshold is poorly suited for the asymmetric cost function. This motivated our investigation into more advanced, cost-aware optimization strategies.

3. Investigating Improvement Strategies

We systematically investigated three diverse strategies to improve upon our simple classifier baseline and minimize the final cost. All experiments were performed on our validation set.

3.1 Strategy 1: Cost-Specific Threshold Optimization

Hypothesis: A uniform 0.5 threshold is suboptimal for an asymmetric cost function. We hypothesized that finding class-specific thresholds that directly minimize the total cost would yield the largest performance gain.

Experiment: We implemented a coordinate descent algorithm. This method iteratively optimizes the decision threshold for each of the 10 classes individually, holding the others constant. For each class, it searches the space of its validation set probabilities to find the threshold that results in the lowest total cost. The process is repeated for a fixed number of iterations or until convergence.

Outcome: This strategy was highly successful. Applying coordinate descent to our "Max Aggregation" pipeline reduced the cost from 55.34 to 34.99. This confirmed that cost-specific tuning is the single most critical step for this task and it became the foundation for all subsequent experiments.

3.2 Strategy 2: Exploring Alternative Aggregation Methods

Hypothesis: The max aggregation function might be overly sensitive to single, spurious high-probability frames. We hypothesized that a more stable aggregator, such as mean or a hybrid function, could lead to a lower final cost after optimization.

Experiment: We implemented and tested two alternative aggregation functions: mean and a weighted hybrid (0.7 \* max + 0.3 \* mean). Each resulting set of segment probabilities was then passed through our full coordinate descent optimization pipeline.

Outcome: This hypothesis was falsified. As shown in Table 1, both Mean Aggregation and Hybrid Aggregation resulted in a higher final cost than the baseline Max Aggregation. We conclude that for our model, sensitivity to the peak probability within a segment was the most effective aggregation strategy.

Strategy Validation Cost

Max Aggregation + Opt. Thresholds 34.99

Hybrid Aggregation + Opt. Thresholds 35.07

Mean Aggregation + Opt. Thresholds 36.29

Table 1: Comparison of final costs for different aggregation methods after full threshold optimization.

3.3 Strategy 3: Post-Processing with Temporal Smoothing

Hypothesis: For acoustically continuous events like a Siren or Chainsaw, the model might incorrectly predict short, intermittent gaps (e.g., a [1, 0, 1] prediction pattern). We hypothesized that a post-processing rule to "fill" these gaps would reduce costly False Negatives more than it would increase False Positive costs.

Experiment: We implemented a vectorized temporal smoothing function that operates on the final binary predictions. For a predefined list of continuous-sound classes (e.g., Siren, Chainsaw, Shout), it identifies [1, 0, 1] patterns within a single audio file and corrects them to [1, 1, 1].

Outcome: This strategy provided a final, marginal improvement. When applied to our best-performing pipeline ("Max Aggregation" with optimized thresholds), it lowered the cost from 34.99 to 34.95. This confirmed that domain-specific heuristics can provide a small but valuable gain. This pipeline became our champion model for the final submission.

4. Real-World Application Viability

Our final system, with a validation cost of 34.95, demonstrates strong performance on the defined task. However, for direct deployment in a real-world application, we would recommend further development.

The primary limitation is that the system is highly overfitted to the specific cost matrix. The optimized thresholds are tailored to a 5:1 or 15:3 FN:FP cost ratio. If a client's priorities were to shift (e.g., making False Positives more critical), the entire threshold optimization process would need to be re-run.

Furthermore, real-world scenarios demand robustness to domain shift, unseen acoustic environments, different microphone hardware, and novel background noises. While our model performs well on this dataset, its performance in a truly novel environment is unknown. Future work should involve testing on a wider variety of data sources and potentially employing data augmentation or domain adaptation techniques to improve generalization.