MLPC 2025 Task 4: Challenge Report

Team Fumbling

Abdalaziz Ayoub **Abdulkarim Al Jamal** **Beibarys Abissatov** **Jeronim Bašić**

Contributions

Abdulkarim Al Jamal handled task 1 (Evaluation Setup) and prepared the report/presentation. Beibarys Abissatov and Abdalaziz Ayoub jointly undertook tasks 2 & 3 (building and improving SED System), with Abdalaziz also tackling task 4 (real-world deployment). Jeronim Bašić focused on the bonus task (pre-trained SED). All team members also contributed to the improvement of strategies for our models.

1 Evaluation Setup

1.1 How did you split the development dataset while avoiding information leakage?

To train and evaluate our multi-label audio event classifier in a robust manner, the shuffled dataset is first divided into 60% training and a 40% temporary set, which is then equally split into validation and test sets, each comprising 20% of the entire dataset. We also partition entire audio files using their unique file identifiers in order to split at the file instead of the frame level, ensuring that no frames from the same file appear in more than one split, which may cause data leakage across the splits.

1.2 Establish and describe a naive baseline system. How much cost can we expect from it?

To establish a naive baseline system (not involving any learned classifier), we implemented a simple majority-vote-based strategy. For each class, we examined the training data and computed whether the class occurred in at least 50% of the time frames. If so, we predicted the class to always be active (output: 1) for all segments in the test set; otherwise, we always predicted it as inactive (output: 0). In our case, none of the classes surpassed the 50% presence threshold, with "Speech" being the most common at 8.6%, and "Rooster Crow" the least common at 0.23%. This led to the system predicted absence for all classes at all time frames, which can perhaps be mitigated with a lower threshold given the sparse dataset; however, we believe this would beat the purpose of a naïve system and therefore kept it as such. This naive approach yielded a balanced accuracy score of 0.5 for every class, as expected when always predicting the negative class in a binary setting with imbalanced labels. However, since it fails to identify any positive events, precision, recall, and F1-score were 0.0 for all classes. More importantly, when evaluated using the challenge-specific cost metric, the system resulted in a total average cost of 108.86 per minute, a relatively high value that reflects the penalties incurred from failing to detect important sound events.

2 SED System

To establish a more robust starting point for sound event detection (SED), we implemented a neural model that combined convolutional and transformer layers. Specifically, the model began with two convolutional (Conv2D) layers, designed to extract high-level time–frequency patterns from the 2D input Mel spectrogram. These convolutional layers preserved both temporal and spectral structure while reducing the input dimensionality. The resulting feature maps were then flattened and passed into a transformer encoder composed of four self-attention layers, each with four attention heads, and a model embedding size of 512. The transformer's role was to model long-range dependencies across the temporal axis and produce rich contextual embeddings for each frame.

This approach was motivated by lessons learned from our previous assignment, in which we had directly flattened the Mel spectrogram and passed it into a transformer. That setup inadvertently collapsed the time dimension and significantly hindered temporal modeling, leading to degraded performance. In contrast, the current convolution-transformer pipeline explicitly preserves and leverages temporal structure, resulting in a more expressive representation.

2.1 How did you threshold and combine predictions to derive a label for the 1.2-second segments? Describe any heuristics or post-processing strategies used.

To derive segment-level labels from frame-level predictions, we applied a simple max-pooling strategy. Specifically, model outputs at the frame level were classified using a fixed decision threshold (0.5), then grouped into non-overlapping 1.2-second segments. Within each segment, we applied max pooling: a segment was assigned a positive label for a class if any frame within that segment was predicted positive for that class. This aggregation approach was implemented via the given *aggregate_targets* function provided in the evaluation code and was applied consistently to both ground truth and predictd labels. We did not apply additional post-processing strategies (e.g., majority voting, smoothing, or sliding windows). Such heuristics could have further reduced false positives or improved temporal coherence, but at the risk of increasing false negatives, which are more costly in our use cases. Therefore, max pooling remained the safest strategy.

2.2 What strategies did you apply to minimize the task-specific cost function?

We adapted the training objective to reflect the asymmetric penalties for false positives (FP) and false negatives (FN) across the ten sound event classes, which was done by computing a custom positive class weight vector (pos_weights) using a ratio of the FN to FP costs. This was then incorporated into the loss function to ensures that, for classes where missed detections are more costly, the model received stronger gradients when it fails to predict the label.

2.3 Does your classifier-based SED system achieve lower costs compared to the naive baseline system?

In terms of performance, our described model achieved a total cost of 81 on the training set. However, it exhibited signs of overfitting and generalized less effectively to the test set, where the cost rose to 101.47. A likely contributor to this overfitting of our initial SED model was the absence of positional encoding within the transformer component. While the convolutional layers capture local structure in the Mel spectrogram, the transformer, which relies heavily on positional information to distinguish between frames occurring at different time points, is thus unable to discern the relative or absolute temporal ordering of audio frames, essentially treating the input as a bag of features. This limits the model's ability to contextualize sound events over time and forces it to memorize training patterns rather than generalizing effectively to new sequences. Consequently, the model performs well on training data but fails to maintain this performance on unseen test samples, as reflected in the increased cost. Nonetheless, it outperformed the naive baseline system, which incurred high false negative costs after failing to recognize any actual sound events.

3 Improvement Strategies

3.1 Describe the main ideas you had and hypotheses you tried as well as their outcomes.

We investigated a wide range of architectural and training strategies aimed at improving the performance of our initial CNN-transformer SED model. For each strategy, we present the underlying hypothesis, a description of the implemented method, the experimental outcome, and our interpretation of the result. All evaluations were conducted on the validation set unless otherwise noted. The experimentation process involved evaluating architectural changes, training strategies, and feature engineering decisions. Through this iterative refinement, we arrived at a out final Sound Event Detection (SED) model which was a lean yet high-performing transformer-based system that significantly outperformed our earlier models. This final model achieved a cost of 37.3 on the validation set and 38.6 on the hidden test set, substantially outperforming all previous configurations.

The following is a non-exhaustive list of the improvement ideas and their corresponding hypotheses & experimental outcomes, as well as an interpretation thereof.

- **Incorporating Positional Encoding:**

- Hypothesis: Transformers require positional encodings to capture temporal structure. Including learnable positional

embeddings will allow the model to learn long-term dependencies and improve sequence modeling accuracy.
- Outcome: Positional encoding led to a significant increase in model performance.
- Interpretation: The transformer architecture lacks inherent awareness of input order. Adding positional encodings provides necessary temporal information, which is particularly crucial in audio tasks where the position of sound events across time matters. Without it, the model struggled to contextualize features across frames.

- **Leveraging Transfer Learning via Pretrained ResNet:**

- Hypothesis: A pretrained ResNet (e.g., ResNet-34), initially trained on ImageNet, can extract useful representations from spectrograms due to visual similarities between images and spectrograms.
- Outcome: The model with the pretrained ResNet backbone outperformed one initialized from scratch.
- Interpretation: Transfer learning allowed the model to benefit from low-level visual filters (e.g., edge detectors) learned from large-scale natural image datasets. These filters generalized well to spectrogram textures, improving learning efficiency and feature richness.

- **Spectrogram Augmentation:**

- Hypothesis: Applying time and frequency masking (as in SpecAugment) increases data diversity and prevents overfitting.
- Outcome: Contrary to expectations, performance degraded.
- Interpretation: The degradation may be due to already limited data per class. Masking further reduced available informative features, hurting class separability. Additionally, our augmentations may have been too aggressive or improperly tuned for this dataset.

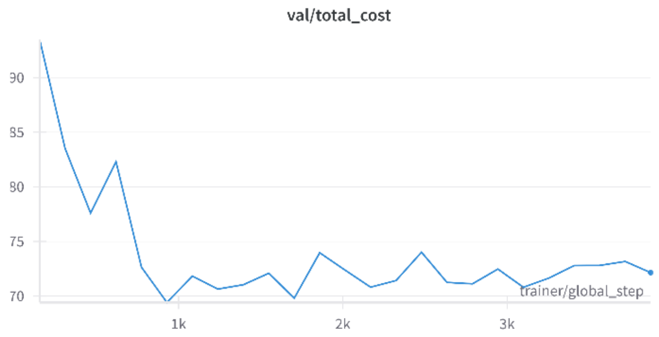


Figure 1: Costs on validation set with spectrogram augmentation

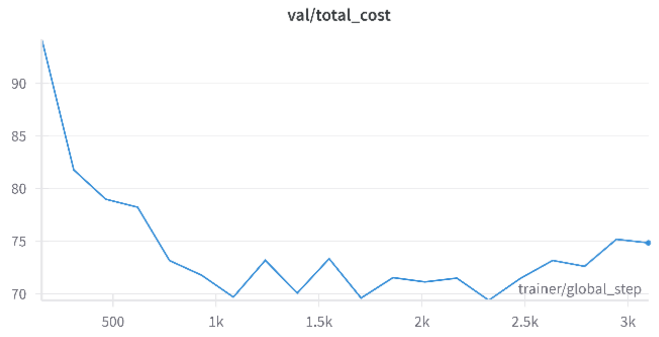


Figure 2: Costs on validation set without spectrogram augmentation

- **Concatenating MFCC Coefficients:**

- Hypothesis: Useful MFCCs (e.g., coefficients 0, 1, 22 from previous report) summarize spectral content and can enrich the mel-spectrogram representation.
- Outcome: No noticeable improvement in performance.

- Interpretation: Although MFCCs theoretically add informative content, concatenating them along the frequency axis without channel separation may have led to suboptimal integration, especially when fed to a pretrained ResNet expecting consistent 2D patterns. A better approach might involve stacking them as an additional channel, although this was incompatible with the Resnet we used.

- **Weight Decay on Learnable Parameters:**

- Hypothesis: Applying weight decay on non-bias and non-normalization parameters will reduce overfitting.
- Outcome: Negligible impact on model performance.
- Interpretation: Overfitting in our case may be driven more by architecture choices (e.g., model capacity, lack of regularization), which simple L2 regularization alone could not resolve.

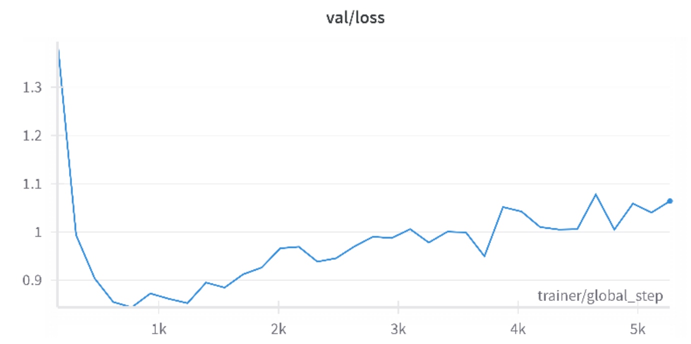


Figure 3: Validation loss with no weight decay

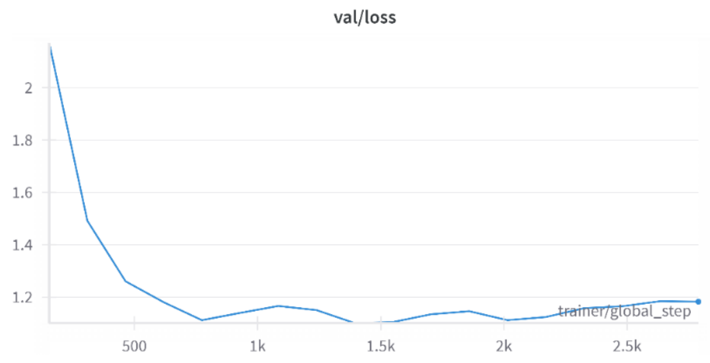


Figure 4: Validation loss with weight decay

- **Cost-Sensitive Tuning of Positive Weights:**

- Hypothesis: Adjusting the class-specific loss weights based on the task-specific cost matrix (pos_weight = FN/FP) should help the model prioritize rare classes or ones with static performance.
- Outcome: Class-specific cost decreased for labels like "Rooster Crow" and "Jackhammer".
- Interpretation: Manually boosting their loss contributions encouraged the model to learn more about them, aligning training dynamics with task priorities.

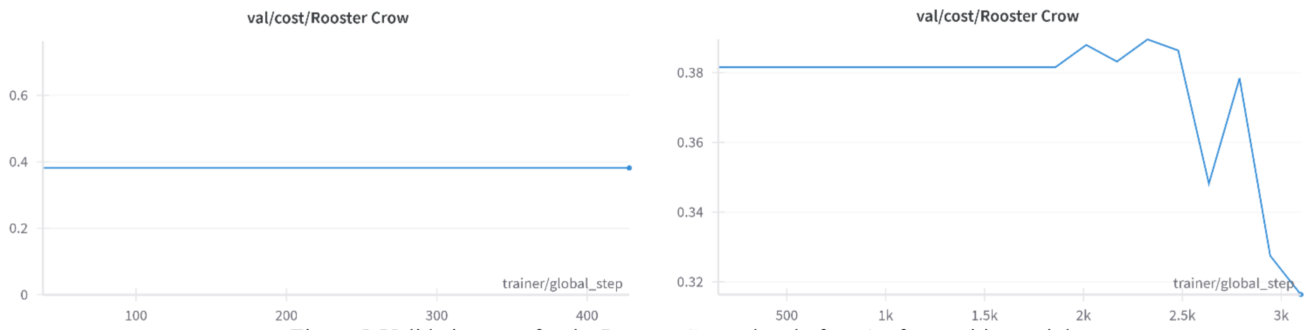


Figure 5: Validation cost for the Rooster Crow class before & after positive weights

- **Using Precomputed Audio Embeddings:**

- Hypothesis: Audio embeddings would underperform compared to raw mel-spectrograms, as state-of-the-art models often prefer raw features.
- Outcome: Surprisingly, models using embeddings consistently outperformed mel/mfcc-based models.
- Interpretation: The embeddings were extracted from a large pretrained SED model and already encode rich, semantically meaningful representations. This made them better suited for downstream classification tasks.

- **Increasing Embedding Size:**

- Hypothesis: Enlarging the transformer embedding size from 512 to 1024 allows the model to capture more complex patterns.
- Outcome: Slight performance increase observed.
- Interpretation: A larger embedding dimension provides more capacity, which can help represent complex data better, although diminishing returns were apparent due to the fixed input complexity and overfitting risk.

- **Adding Additional Linear Layers:**

- Hypothesis: A deeper classification head with two linear layers will improve the transformation of learned embeddings into class probabilities.
- Outcome: Minor performance gains.
- Interpretation: The additional layer may help with learning non-linear decision boundaries but was not transformative. The base model likely already had sufficient capacity.

- **Increasing Attention Heads:**

- Hypothesis: Increasing the number of attention heads in the transformer from 4 to 6 enables the model to attend to a greater diversity of subspaces, improving performance.
- Outcome: No measurable impact on results.
- Interpretation: The model may have already been attending to sufficient aspects of the sequence. Adding heads without changing embedding size likely diluted per-head capacity.

- **Replacing CNN with Linear Projection:**

- Hypothesis: Pretrained audio embeddings are rich and frozen; therefore, a simple linear projection instead of a CNN is sufficient and more efficient.
- Outcome: This approach outperformed all previous ResNet-Transformer models and was used in the final model. Interpretation: The frozen embeddings require minimal transformation. Replacing the CNN with a linear layer avoids unnecessary parameter overhead and preserves the integrity of the learned representation. It also enables faster and more stable training.

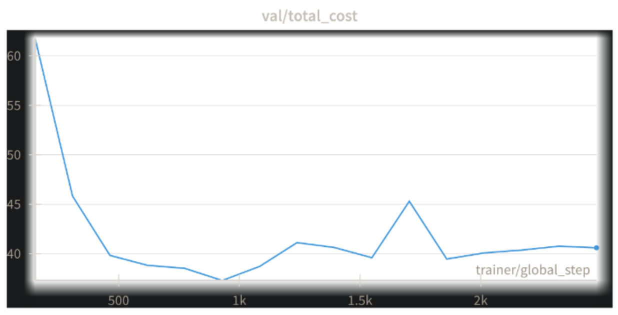
4 Real-World Applicability

4.1 Do you think your final system could be deployed in a real-world application? In your opinion, which aspects of the project or your system would need to be adapted to fulfill possible real-world requirements?

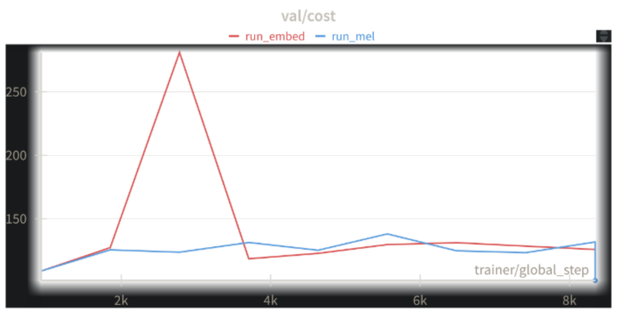
Our final sound event detection system demonstrates strong performance. However, its estimated model parameters size of 163 MB makes it unsuitable for ultra-low-power devices like hearing aids or environmental sensors. To make it suitable for real-time or embedded applications in urban environments, we must compress the model via quantization/distillation. Furthermore, our final system's transformer-based model relies on precomputed audio embeddings generated by a separate, pretrained sound event detection model for training. In practical applications, it is essential that the model directly operates on raw audio features (e.g., mel-spectrograms, MFCCs) rather than relying on intermediate embeddings from an external model. The use of frozen embeddings adds a dependency on a second, potentially large model that must either be bundled with the system or queried during inference, increasing computational overhead, latency, and power consumption, all of which are problematic for low-power devices or real-time applications.

5 Bonus

5.1 Compare the performance of the resulting system your best model.



(a) Baseline Transformer
40.8M parameters
Final cost: 37.299 (epoch 16)



(b) ATST-Frame
88.0M parameters
Early stop: 101.474 (epoch 27)

Figure 6: Validation costs convergence comparison showing the baseline table optimization vs the ATST-frames volatile training

The baseline transformer model, with its efficient 40.8 million parameters, demonstrated significantly better performance than the pretrained ATST-Frame model containing 88.0 million parameters. This was evidenced by the baseline's superior validation cost of 37.299 achieved by epoch 15, compared to the ATST-Frame's validation cost of 101.474 that led to early stopping at just 27 epochs (Figure 6).

First, the baseline's architectural efficiency - using 2.2 times fewer parameters than the ATST-Frame - enabled faster convergence, improving from an initial validation cost of 55.2 in the first epoch down to its optimal 37.299.

Second, the baseline's processing of fixed embeddings proved more stable for the 1.2 second segmentation task compared to the ATST-Frame's end-to-end spectrogram processing, which exhibited volatile per-class costs during training.

Third, while the baseline achieved optimal performance by epoch 16, the ATSTFrame's training was halted early due to its inability to escape high-cost regimes, suggesting it required either more training data or extended adaptation time.

For this specific urban noise monitoring application, where mechanical sounds like Chainsaw (achieving F1 score of 0.801 with cost of 1.233) and Siren (F1 score of 0.772 with cost of 4.023) were particularly important, the smaller baseline model demonstrated clear advantages in both training stability and final cost optimization.

However, the ATST-Frame's spectrogram-level feature learning and sophisticated temporal attention mechanisms remain theoretically better suited for applications requiring detection of novel sound classes not present in the training set or needing temporal resolution finer than the current 1.2 second segments. Future improvements could explore hybrid architectures combining the ATST-Frame's superior feature extraction with the baseline's cost-optimized classification head, particularly focusing on improving performance for high-cost classes like Horn Honk (current recall of 0.679 but cost 1 of 7.508) and Jackhammer (cost of 4.156 despite moderate performance).