Sound Event Filtering and Classification

Bingjie Wang

December 15, 2025

Introduction

Sound Event Detection (SED) is widely used in:

- Smart home monitoring (glass break / alarms)
- Surveillance and public safety
- Robotics and autonomous navigation
- Video understnding (Smart subtitle)

Challenge: Real-time SED requires:

- Low latency (no future frames)
- Efficient models (edge devices)
- Dynamic adaptation to acoustic complexity

Problems with Existing Methods

CNN/CRNN SED

- Require fixed input window (e.g., 100 frames)
- Latency increases as window grows

Transformer SED

- Need global context $\rightarrow$ cannot operate in real-time
- High computational cost

None supports adaptive frame window length.

Project Goal

We propose a SED model with:

- **Adaptive Frame Selector** (learned from audio content)
- **Lightweight Transformer Encoder** (¡1M parameters)
- **Masked-Frame Training** to simulate streaming constraints

Target benefits:

- Reduce latency by shortening frames when possible
- Maintain accuracy using Transformer modeling power
- Provide real-time, causal prediction

Adaptive Frame Selector (Transformer-based)

Given a Mel-spectrogram $M \in \mathbb{R}^{T \times F}$, we first map each frame to an embedding:

$$e_t = W_{\text{emb}} M_{t,:} \in \mathbb{R}^d, \quad t = 1, \ldots, T$$

and obtain a sequence $E = [e_1, \ldots, e_T]$.

We pass E through a lightweight Transformer encoder:

$$H = \text{TransformerEncoder}(E) \in \mathbb{R}^{T \times d}$$

Then we compute a scalar importance score for each frame:

$$s_t = q^\top h_t, \quad \alpha_t = \frac{\exp(s_t)}{\sum_{j=1}^{T} \exp(s_j)}$$

We define the adaptive prefix length T_{new} as the smallest k such that

$$\sum_{t=1}^{k} \alpha_t \geq \rho,$$

where $\rho \in (0, 1]$ is a coverage threshold (e.g., 0.9).

Masked Frame Training (MFT)

We simulate streaming not by simple tail masking, but by creating *time-progressive pairs* of audio segments.

For each recording, we sample two segments:

$$x^{(a)} = M_{1:k}, \quad x^{(b)} = M_{1:k+\Delta},$$

where k is a base length and Δ is a small time extension (e.g., 50–100 frames) that simulates the next step in the stream.

Two-stage training:

- **Stage 1 (fixed window):** train the Transformer SED model on segments of predefined length (e.g., $k = 100$) so that it can already detect events in a local window.
- **Stage 2 (adaptive + pairs):** plug in the adaptive frame length selector, use its predicted length $\hat{k}$ to construct pairs $(x^{(a)}, x^{(b)})$ and jointly train selector + SED model to make consistent predictions as time progresses.

Transformer Architecture

- 2 encoder layers
- 4 attention heads
- Embedding dim $= 128$
- FFN hidden dim $= 256$

Total parameters approx. 0.83M

Latency per 40-frame input approx. 2.1 ms on RTX 4090D

Classification Subnetwork

After the adaptive Transformer encoder, we obtain a sequence of frame-level embeddings:

$$H = [h_1, \ldots, h_{T_{\text{new}}}] \in \mathbb{R}^{T_{\text{new}} \times d}.$$

Following open-source audio tagging models (e.g., CNN14 / YAMNet), we use a small classification subnetwork:

$$z = \left[\text{mean}_t(h_t) \parallel \max_t(h_t)\right] \in \mathbb{R}^{2d},$$

$$z' = \phi(W_2\,\text{ReLU}(W_1 z)), \quad \hat{y} = \text{softmax}(W_{\text{cls}} z').$$

Notes:

- Structure of the head is inspired by open-source projects (PANNs / YAMNet): global pooling + 2-layer MLP.
- This separates temporal modeling (Transformer) from clip-level decision making (classification head).

Dataset

UrbanSound8K (Kaggle mirror) 10-class sound event dataset.

- 8732 audio clips
- Sampling rate 44.1 kHz
- 40 Mel bins

$Train/Val/Test = 80/10/10 split$.

Evaluation Metrics

Classification (UrbanSound8K):

- Accuracy
- Macro F1-score

Efficiency:

- Latency per frame (ms)
- GFLOPs per inference
- Parameter count

Experimental Settings

- Optimizer: Adam, lr = 1e-4
- Batch size: 16
- Training epochs: 100
- Mask probability: 0.3

Baselines:

- CNN baseline (4 conv layers)
- CRNN (CNN + GRU)
- Our model w/o adaptive selector
- Final full model

Results: Accuracy Comparison

Model	Acc	F1	Params
CNN Baseline	78.2	76.5	1.6M
CRNN	82.7	81.9	1.9M
Ours (no adaptive)	83.1	82.0	0.83M
Ours (full)	**84.4**	**83.2**	0.83M

Efficiency Results

Model	Latency (ms)	GFLOPs	Avg Frames
CNN Baseline	6.5	1.8	200
CRNN	8.1	2.3	200
Ours (no adaptive)	3.4	0.48	200
Ours (adaptive)	**2.9**	**0.39**	**120**

Overall latency improvement: 15% over non-adaptive model, 28% over CRNN.

Thank You