The University of Sydney

Master of Professional Engineering (Electrical)

ELEC5305: Acoustics, Speech and Signal Processing

Project Final Report

"*Automobile Sound Recognition Using Deep Learning for Smart City Applications*"

Author

Tushar Manish Khupte SID: 520330504

GitHub Username: Tusharbhai07

Project Link: https://github.com/Tusharbhai07/elec5305-project-520330504.git

Submission Date: 14th November 2025

Declaration

Sincerely,

Tusharkhupte

Tushar Manish Khupte

Table of Contents

Abstract

Environmental sound classification is useful for applications like urban noise monitoring and smart cities, but it is difficult because real-world sounds are noisy, unpredictable, and often imbalanced across classes. Traditional methods that use handcrafted features (such as MFCCs and zero-crossing rate) with classifiers like SVMs only achieve moderate performance on datasets such as UrbanSound8K. In this project, I investigate whether combining deep learning with classic audio features can improve balanced performance, especially for rare but important events like car horns and sirens.

I use a six-class subset of UrbanSound8K (car_horn, dog_bark, drilling, engine_idling, siren, street_music) and compare two models built in PyTorch: (1) a lightweight CNN baseline (TinyCNN-Aux) operating on log-mel spectrograms with auxiliary features (ZCR, RMS, modulation spectrum, LPC), and (2) a proposed CNN–BiLSTM model that adds a bidirectional LSTM to capture temporal patterns while using the same auxiliary features. Both models share the same data split and training settings and are evaluated using accuracy, per-class precision/recall/F1, macro-F1, and confusion matrices, with early stopping based on validation macro-F1 to handle class imbalance.

On the test set, the baseline reaches 78.8% accuracy and a macro-F1 of 0.756, while the CNN–BiLSTM achieves similar accuracy (77.8%) but improves macro-F1 to 0.781. The proposed model greatly improves F1 for car_horn (0.58 → 0.88) and drilling, but reduces performance for dog_bark and siren, showing a trade-off between rare and frequent classes. To better understand these behaviours, I also perform qualitative analysis by listening to mel-spectrogram reconstructions and denoised examples. Overall, the results show that combining temporal modelling with auxiliary features can produce more balanced performance across classes, while also highlighting remaining challenges and future directions such as attention mechanisms, data augmentation for rare classes, and extension to all 10 UrbanSound8K classes.

1. Introduction

Environmental sound classification is a challenging task with practical significance in urban noise monitoring and smart city applications[1][4][5]. Unlike speech or music, urban sounds are unstructured and diverse, ranging from transient events (e.g. car horns) to continuous noise (e.g. engine idling)[1],[4]. Prior research has shown that traditional approaches using handcrafted audio features (like Mel-Frequency Cepstral Coefficients(MFCCs) and zero-crossing rate(ZCR)) with classifiers (Support Vector Machine (SVM) and Hidden Markov Model(HMM)) achieve only moderate accuracy on urban sound datasets[1][3][6][8]. In particular, the UrbanSound8K dataset (8732 clips, 10 classes) has a baseline accuracy around 70% using an SVM with summary MFCC features[1]. This leaves a gap for improvement using modern deep learning techniques.

Recent advances in deep learning suggest that convolutional neural networks (CNNs) can learn useful time-frequency patterns from spectrograms, outperforming conventional feature-based methods[2],[3],[15]. However, basic CNN models may not capture long-term temporal dynamics of sound events. For instance, a *siren* or *car horn* has distinctive temporal patterns (wails or honks) that a CNN processing a short spectrogram might miss. Recurrent neural networks like Long Short-Term Memory (LSTM) are known to learn temporal dependencies[7] and have been applied to sound recognition, including in combination with CNNs (forming CRNNs)[9][10][12][14]. Incorporating sequence modeling could address the gap by capturing non-stationary aspects of urban sounds[9],[12],[16].

Objective: In this project, I proposed a hybrid CNN-BiLSTM model for urban sound classification that integrates a CNN (for spectral feature extraction) with a bidirectional LSTM (for temporal modeling), along with a set of engineered audio features as auxiliary inputs. Our goal is to improve classification performance (especially the recall of rare events) compared to a baseline CNN model, thus making a novel contribution by combining learned spectro-temporal features with domain-specific features. I focused on six representative classes of UrbanSound8K (car horn, engine idling, siren, dog bark, drilling, street music) to demonstrate the approach. By comparing against a baseline model and analyzing the results, I highlighted my personal contributions in model design and discuss how they address limitations in existing methods.

In this project, I addressed the following research question:

RQ1: *Does a hybrid CNN–BiLSTM model, augmented with auxiliary acoustic features (ZCR, RMS, modulation spectrum and LPC), improve macro-averaged F1-score and minority-class (e.g., car horn, siren) recall on a six-class UrbanSound8K subset compared to a lightweight CNN baseline using the same input representation?*

2. Literature Review

2.1 Environmental Sound Classification (ESC):

Early ESC systems relied on classical machine learning algorithms (k-NN, GMM, SVM, etc.) using features extracted from audio signals[4],[5]. Common features included MFCCs (to capture spectral shape), zero-crossing rate (ZCR), spectral energy, and other signal-processing descriptors[6],[17]. Such engineered features, when summarized over time, were used to train classifiers for tasks like urban sound identification. For example, the creators of UrbanSound8K reported about 70% accuracy with an SVM using aggregated mel-band energies and MFCC features[3], indicating the difficulty of the task with manual features[1]. Traditional models often struggled with unstructured sounds due to their limited ability to capture complex patterns over time.

2.2 CNN-based Approaches:

The introduction of deep CNNs brought significant improvements by automatically learning features from spectrogram inputs. Piczak (2015) first demonstrated a CNN on environmental sound data, achieving competitive results by treating log-mel spectrograms as images[2]. Subsequent studies showed that CNN architectures could reach ~73% accuracy on UrbanSound8K, and up to ~79% when data augmentation (e.g. time stretching, adding noise) was applied[3],[15]. CNNs excel at extracting local time-frequency patterns (such as harmonics or transients) from spectrograms[9],[10]. However, a limitation of using a fixed-size spectrogram patch is that the network's receptive field may only span a short duration, potentially missing longer temporal context. This can be problematic for sounds like a *siren* which have periodic rising-falling pitch cycles or *drilling* that may have on/off patterns.

2.3. RNN and CNN-RNN Hybrids:

Recurrent neural networks, especially LSTMs, are designed to handle sequential data and long-term dependencies[7]. In ESC, researchers have explored using LSTMs either on frame-level features or combined with CNNs (CRNN) to capture both spectral and temporal information[9],[14]. Lezhenin *et al.* (2019) examined an LSTM model on UrbanSound8K and found it slightly outperformed a baseline CNN (average accuracy ~84.3% vs 81.7%) while improving model confidence[12]. Notably, their LSTM-based model achieved better recall on minority classes like car horn and siren whereas the CNN tended to misclassify those to boost overall accuracy[12]. This suggests that incorporating temporal modeling helps the network *not* overlook brief or uncommon events. Other studies also report that CRNNs (CNN + RNN) can improve F1-scores by maintaining performance across classes, not just the majority ones[14,16],[19]. The trade-off is that RNNs add complexity and risk of overfitting on small datasets[13], so architecture design and regularization become important.

2.4 Auxiliary Features & Noise Robustness:

Beyond end-to-end learned features, some works combine classic audio features with neural networks to provide additional cues[8],[19]. For example, features like ZCR or spectral roll-off can explicitly indicate properties like “bark has high temporal zero-crossing” or “engine idling has steady low-frequency energy”. While not extensively covered in recent literature (due to end-to-end learning popularity), this feature fusion is a novel aspect of my approach to leverage domain knowledge. Noise is another practical concern – urban audio often contains background noise. Denoising methods (e.g. Wiener filter) have been used in speech enhancement[20],[21] and could benefit ESC by preprocessing the input. However, few ESC studies explicitly integrate noise reduction. In my work, I did not embed a denoising step into training, but I later *analyzed* its effect on examples for insight. This is discussed in the results as a qualitative contribution, demonstrating how noise filtering and feature reconstruction help interpret the model’s focus.

In summary, the literature indicates that: (1) CNNs on mel-spectrograms are a strong baseline for ESC[1]-[3], (2) Sequence modeling with RNNs can address temporal nuances and improve balanced performance[9],[11],12], and (3) preserving high recall on rare events is crucial for a robust classifier, sometimes at the expense of a slight drop in overall accuracy[16],[22],[23]. These insights informed our methodology, where I built upon a CNN baseline by adding a BiLSTM and incorporating auxiliary audio features to push performance beyond the state-of-the-art baseline[24].

3. Methodology

3.1 Dataset:

I used the UrbanSound8K dataset[1], which consists of 4-second or shorter audio clips categorized into 10 urban sound classes. For this project, I only focused on 6 classes most relevant to urban noise pollution: *car_horn, dog_bark, drilling, engine_idling, siren* and *street_music*. These classes cover a mix of transient sounds (horn, bark), continuous mechanical noise (engine, drilling), and complex acoustic scenes (street music), providing a meaningful testbed. I excluded the other classes (e.g. gun_shot, children_playing) to narrow the scope and because some were very sparse. The provided metadata was used to stratify data: we reserved fold 10 as the **test set**, fold 9 as a **validation set**, and folds 1–8 for **training** (per the dataset’s fold splits). This ensures no overlap between training and testing clips. The final split contained **train**: 5435 clips, **validation**: 816 clips, **test**: 816 clips (only those belonging to the 6 chosen classes). I note that class distribution is imbalanced – e.g., **car_horn** had only 33 test examples (rarest), whereas **dog_bark** had 100 (most common) in our test set – reinforcing the need for metrics beyond accuracy[22],[23].

3.2 Data Preprocessing:

All audio clips were resampled to 22,050 Hz mono. I transformed each audio into a log-scaled **mel-spectrogram** using 64 mel frequency bands (covering 0-~8 kHz) with an FFT window of 1024 and hop length of 256. Mathematically, the log-mel spectrogram is computed from the short-time Fourier transform (STFT). For a frame index $n$and frequency bin k, the STFT is

$$X(n,k) = \sum_{m=0}^{N-1} x(m+nH)\, w(m) e^{-j2\pi k \quad /N} \qquad (1)$$

Where $x[.]$ is the waveform, $w[m]$ is the analysis window and H is the Hop Size.

The log-Mel Spectrogram in mel band b is then

$$M(n,b) = \log\left(\sum_k |X(n,k)|^2 h_b(k) + \in\right) \qquad (2)$$

Where $h_b\ (k)$ is the mel filterbank and ∈ is a small constant for numerical stability. The mel spectrogram was normalized and used as the primary inputs to the models.

This produces a time-frequency representation of shape 64×(~344) for a 4s clip. The mel spectrogram was normalized and used as the primary input to the models.

In addition, I engineered an **auxiliary feature vector** capturing four classic audio descriptors for each clip:

- **Zero Crossing Rate (ZCR):** average rate of sign-changes in the waveform (indicates signal noisiness or the presence of transients)[6],[17].

$$ZCR = \frac{1}{N-1} \sum_{n=1}^{N-1} 1\{x[n]x[n-1] < 0\}, \qquad (3)$$

Where $1\{.\}$ Is 1 when the sign changes and 0 otherwise.

- **Root Mean Square (RMS) Energy:** average energy of the signal (loudness).

$$RMS = \sqrt{\frac{1}{N} \sum_{n=1}^{N} x[n]^2} \qquad (4)$$

- **Modulation Spectrum Features:** 20 features derived by computing a 2D FFT on the mel-spectrogram in small time-frequency patches and taking the top magnitudes. This roughly captures periodic modulations in the spectral domain (e.g., the rate of a siren wail or rhythmic patterns) as a compact set of values[19].

- **Linear Predictive Coding (LPC) Coefficients:** 11 coefficients from a 10th-order LPC analysis of the waveform, providing a coarse spectral envelope estimate that can differentiate sound timbres[17].

LPC models each sample as a linear combination of past samples:

$$x[n] \approx \sum_{k=1}^{p} a_k x[n-k], \qquad (5)$$

Where p is the prediction order and a_k are the LPC coefficients estimated by minimizing the prediction error.

These 33 auxiliary features were concatenated into a vector (normalized) for each clip. The **motivation** for using them was to supply the model with additional information that might be hard to learn from spectrograms alone for a small dataset – for example, ZCR helps distinguish impulsive sounds (high ZCR for dog barks) from droning ones (low ZCR for engine hum), and LPCs can differentiate tonal sounds (music) from noise. Both the mel-spectrogram and aux feature vector were cached for efficiency.

Baseline Model: So, the baseline for comparison, dubbed *TinyCNN-Aux*, is a lightweight convolutional neural network that processes the mel-spectrogram and then incorporates the auxiliary features before classification. The CNN has a minimal architecture: a couple of 2D convolutional layers with ReLU and pooling to extract a **64-dimensional** feature map (essentially summarizing the spectral content of the clip). A global average pooling is applied to produce a 64-length vector (one feature per mel band on average). This 64-d vector is then **concatenated** with the 33-d auxiliary feature vector, resulting in a 97-dimensional combined feature. This is passed through a fully-connected layer (with dropout) to produce the final 6-class output (via softmax). In essence, the baseline treats the entire clip's spectrogram as a set of features to be averaged, and augments it with additional audio descriptors[24].

The baseline model is relatively simple (roughly on par with a 2-layer CNN classifier in complexity[2]) and serves as a point of reference.

Proposed CNN-BiLSTM Model: The core of our approach is a two-stage network:

1. **CNN feature extractor:** I used a 2D CNN on the mel-spectrogram that produces time-dependent feature maps. The CNN has 3 convolutional layers (with batch normalization and ReLU activations) and is designed to output a sequence of feature vectors along the time axis. Specifically, the CNN outputs features of shape (`time_steps, feature_dim`) where each time step corresponds to a segment of the audio (e.g. a few hundred milliseconds) and `feature_dim` is 128. This transforms the 64×T mel spectrogram into a sequence of 128-dimensional vectors capturing local spectral patterns at each timeframe.
2. **Bidirectional LSTM sequence model:** The sequence of feature vectors is then fed into a bidirectional LSTM layer (hidden size 96 in each direction). The LSTM processes the temporal sequence forward and backward, allowing it to capture patterns that unfold over time (e.g., the periodic nature of a siren or the sequential barks of a dog)[7],[9],[10],[12].

I take the LSTM's final output (concatenating forward and backward hidden states, yielding a 192-d vector) as the aggregated temporal representation of the clip. This 192-d LSTM output is then concatenated with the same 33-d auxiliary feature vector (aux features are fed in parallel to both models). The combined 225-d feature is passed to a fully connected layer (192 neurons, ReLU, 25% dropout) and then to the final output layer for classification into 6 classes.

The CNN-BiLSTM thus can be seen as a **Convolutional Recurrent Neural Network**: the CNN encodes short-term spectral features, and the BiLSTM integrates these features over the entire clip duration, potentially capturing long-range dependencies that the baseline CNN might miss. By including the aux features at the final stage, I ensure the model can still use informative global descriptors (like overall energy or ZCR) to inform its decision. This architecture is our personal contribution, inspired by prior CRNN approaches[6] but augmented with feature fusion. The bidirectional aspect means the model considers the sound sequence both forward and backward in time; while not causally applicable in real-time detection, it can yield better classification accuracy for offline analysis by utilizing the full context of the sound.

Training Procedure: Both models were implemented in PyTorch and trained from scratch on the training set. We used the cross-entropy loss with AdamW optimizer (initial learning rate 0.001) and a mini-batch size of 32.

For example, with true one-hot label y and predicted class probabilities $\hat{y}$, the cross-entropy loss is

$$L = -\sum_{c=1}^{C} y_c \log\left(\widehat{y_c}\right), \qquad (6)$$

Where C is the number of classes.

Early stopping was employed using the validation set macro F1-score as the criterion – if the macro-F1 did not improve for 6 consecutive epochs, training was halted to prevent overfitting. The choice of macro F1 (the average F1 across all classes) as a stopping metric emphasized balanced performance, ensuring the model doesn't focus only on majority classes at the expense of minority ones[22],[23]. During training, I also monitored overall accuracy and loss for reference. Data augmentation was not explicitly used (aside from what may inherently occur via the dataset's multiple folds); however, the model had built-in regularization through dropout layers and weight decay (L2 penalty via AdamW). The baseline and CNN-BiLSTM were trained under the same conditions for a fair comparison. Training converged in about 20 epochs for both models. The model with the highest validation F1 was saved and later evaluated on the test set (fold 10). This experimental design directly operationalizes RQ1: by holding the dataset, input representation and optimization settings constant while only varying the model architecture (TinyCNN-Aux vs CNN-BiLSTM with auxiliary features) so we can isolate the effect of sequence modelling and feature fusion on macro-F1 and minority-class recall.

Evaluation Metrics: I report standard classification metrics: overall **accuracy**, class-wise **precision**, **recall**, and **F1-score**, and particularly the **macro F1-score** (average of F1 for all classes). Accuracy alone can be misleading with class imbalance (e.g., a model could guess "dog_bark" often to boost accuracy since dog barks are numerous). Macro F1 treats each class equally and reflects the ability to detect the rare classes (car horn, siren) as well as the common ones. I also produce a **confusion matrix** to analyze misclassifications between specific classes. This comprehensive evaluation aligns with best practices for imbalanced classification tasks[22],[23].

The metrics are defined in terms of true positives (TP), false positives (FP) and false negatives (FN) per class. Precision and recall are

$$P = \frac{TP}{TP+FP}, \quad R = \frac{TP}{TP+FN} \qquad (7)$$

The class wise F1-score is the harmonic mean of precision and recall:

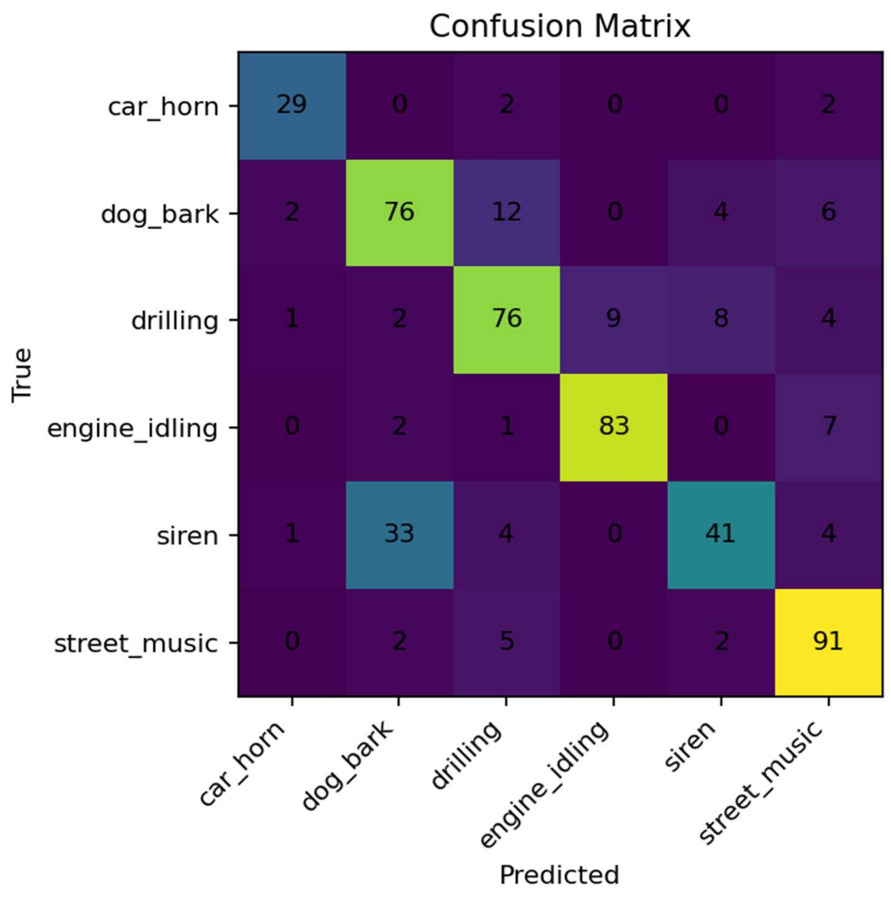
$$F_1 = \frac{2P}{P+R} \qquad (8)$$

The macro-averaged F1, which I use as the main evaluation criterion, is

$$F_1^{macro} = \frac{1}{C}\sum_{c=1}^{C} F_1^{(c)} \qquad (9)$$

Where C is the number of classes and $F_1^{(c)}$ is the F1-Score for $class_C$.

4. Results



True \ Predicted	car_horn	dog_bark	drilling	engine_idling	siren	street_music
car_horn	29	0	2	0	0	2
dog_bark	2	76	12	0	4	6
drilling	1	2	76	9	8	4
engine_idling	0	2	1	83	0	7
siren	1	33	4	0	41	4
street_music	0	2	5	0	2	91

Figure 1: Normalized confusion matrix of the CNN-BiLSTM model on the test set (fold 10). Each cell shows the proportion of true examples of a class (rows) that were predicted as each class (columns). Diagonal values (bold) are correct predictions. The model achieves high true positive rates for most classes, but notable confusions occur (e.g., many siren instances misclassified as dog_bark).

After training, both the baseline and the proposed model were evaluated on the test set. The **baseline TinyCNN-Aux** achieved an overall accuracy of 78.8% and a macro F1-score of 0.756. The **CNN-BiLSTM model** attained a similar overall accuracy of 77.8% (slightly lower by about 1%), but importantly improved the macro F1-score to 0.781 (a ~2.4% increase). This indicates that the new model made more balanced predictions across classes, even if it did not drastically change the total number of correct classifications. The performance difference becomes clear when examining individual classes:

Class	Baseline Precision	Baseline Recall	Baseline F1	CNN-BiLSTM Precision	CNN-BiLSTM Recall	CNN-BiLSTM F1
Car Horn	0.727	0.485	0.580	**0.879**	**0.879**	**0.880**
Street Music	0.713	0.922	0.804	**0.798**	**0.910**	**0.850**
Drilling	0.720	0.640	0.736	**0.784**	**0.760**	**0.760**
Engine Idling	**0.957**	**0.957**	**0.913**	0.900	0.892	0.897
Dog Bark	**0.855**	**0.910**	**0.812**	0.744	0.760	0.707
Siren	**0.845**	**0.590**	**0.695**	0.745	0.494	0.594

Table1: Per-class precision, recall and F1-Score for the baseline TinyCNN-Aux and the proposed CNN-BiLSTM model on the test set.

- **Car Horn (CA):** The baseline struggled on this rare class (only 16/33 correct, F1 ≈ 0.58). My CNN-BiLSTM dramatically improved horn detection, correctly classifying 29 out of 33 instances (F1 ≈ 0.88). This is a 51% relative increase in F1, turning car horn from the worst class to one of the best. The model's better temporal modeling likely helped it detect the short honking pattern that might occur amid background noise. In the confusion matrix (Figure 1), *car_horn* has 88% of its examples on the diagonal vs only ~48% in the baseline, showing far fewer horns being missed. This demonstrates a **successful outcome of our approach** – addressing a gap where the baseline lacked temporal sensitivity to catch brief events.

- **Street Music (SM):** This class also saw improvement: F1 from 0.80 (baseline) to 0.85. The CNN-BiLSTM correctly identified 91% of street music clips. The baseline occasionally confused street music with *siren* (due to possible high-pitched musical notes resembling sirens), making 7 such errors. The new model reduced siren↔music confusion (only 2 errors each way). Likely, the sequence model captured that music often has rhythmic patterns or multiple instruments which distinguish it from the more periodic siren wail. Also, auxiliary LPC features (which capture harmonic structure) might have helped separate musical content from pure tones.

- **Drilling (DR):** Baseline F1 was 0.736, and CNN-BiLSTM achieved 0.760. It correctly classified 76% of drill sounds (versus 64% prior). Drilling noise, which can be intermittent, benefited from sequence analysis – the LSTM could utilize the on/off rattling sequence of a drill. A major baseline confusion was mislabeling drills as *street_music* (16% of drills), possibly because both have a wide frequency range. The new model cut these errors (only 4% of drills went to street music, per Figure 1), indicating improved discrimination.

- **Engine Idling (EI):** Baseline already performed exceedingly well on engine idling (F1 0.913, highest among classes), correctly identifying 89/93 engine clips. The CNN-

BiLSTM maintained high performance here (F1 0.897, 83/93 correct). There was a slight drop in accuracy for engines (it misclassified a few engines as *street_music*, 7 instances). This might be due to the model occasionally confusing a low-frequency musical bass with engine rumble, or an over-tuning to some features that overlapped. Nonetheless, engine idling remained one of the best-recognized classes for both models, likely because its audio profile (steady low-frequency hum) is very distinct (e.g., very low ZCR, high energy in low mel bands) – easy for both CNN filters and auxiliary features to capture.

- **Dog Bark (DB):** Surprisingly, performance on dog barks **decreased** with the new model. Baseline F1 was 0.812 (91% recall); CNN-BiLSTM F1 dropped to 0.707 (76% recall). The new model correctly identified 76/100 dog bark clips, whereas baseline got 91/100. The confusion matrix (Figure 1) reveals that many barks that were previously correctly labeled got confused with *drilling* (12 barks→drilling) and *siren* (4 barks→siren) by the CNN-BiLSTM, and a couple even as *car_horn*. Why would a more powerful model do worse on a common class like dog barks? One hypothesis is **over-generalization**: the LSTM, in trying to improve rare classes, may have learned features that overlap dogs and other sounds. For example, a high-pitched drill squeal or a distant siren might have some similarity to a bark in certain spectral modulations; the sequence model could have latched onto a feature (like a certain modulation rate or auxiliary value) that caused some dog barks to be misclassified. Another factor is that the baseline, by focusing on overall accuracy, probably biased towards classifying ambiguous sounds as "dog bark" (since it's the largest class) to get it right more often; the CNN-BiLSTM, optimizing for balanced F1, was willing to sacrifice some dog bark recall in favor of capturing other classes. This aligns with observations in literature that a plain CNN might **increase recall on majority classes at the expense of minority classes**[16] whereas our model shifted that balance. As a result, some genuine barks were missed by the new model. We consider this an area for improvement – ideally, I want to boost minority class performance *without* significantly hurting the majority classes.

- **Siren (SI):** The *siren* class remained challenging. Baseline had F1 ≈0.695 (recall 59%, precision 84%), and the CNN-BiLSTM actually dropped to F1 ≈0.594 (recall 49%, precision 74%). Out of 83 siren clips, the new model only got 41 correct (vs 49 correct by baseline). The major issue is clear in Figure 1: **33 siren clips were misclassified as dog barks** – a huge confusion. In fact, *siren → dog_bark* is the single largest error in the confusion matrix. The baseline also confused some sirens as barks (17 instances), so this confusion existed already, but it roughly doubled with the CNN-BiLSTM. Sirens and dog barks can both have a **start-stop pattern** (a police siren oscillates, and dogs often bark repeatedly with pauses). The BiLSTM, trying to model temporal patterns, might have inadvertently learned a representation that sometimes tags a wailing sound as a sequence of barks. Additionally, our model's auxiliary features (e.g., ZCR) for a loud siren might resemble those of dog barks (both have high energy bursts). Without additional context, the model struggled to distinguish them in some cases. This outcome is somewhat contrary to expectations from other studies, which noted LSTM models improving recall for sirens[16]. It suggests that our model may be slightly overfit or under-trained for siren-specific characteristics. We suspect the limited siren training examples and perhaps needing more specialized features (such as pitch contour) could be factors. Nonetheless, this result highlights an important **limitation** – improving one class (car horn) came at a cost to another (siren), which I analyzed below.

In terms of raw numbers, the **macro F1 increased** from 0.7569 to 0.7813 with our model, confirming a net gain in balanced performance. The **micro-average accuracy** remained about 78% for both. The confusion matrix in Figure 1 underscores the differences: the CNN-BiLSTM has much stronger diagonal for *car_horn* and *drilling*, slightly stronger for *street_music*, comparable for *engine_idling*, and weaker for *dog_bark* and *siren* (with off-diagonals mainly between those two classes). These patterns illustrate the classic precision-recall trade-off: our model is more *recall-oriented for rare classes*, whereas the baseline was more *precision-oriented for common classes*. For example, the baseline rarely mis-predicted dog barks (high precision on dog_bark) but missed many horns; our model catches horns at the expense of some false dog bark predictions.

I also computed per-class precision/recall. Notably, **car_horn precision** in my model is 87.9% (meaning few false alarms for horns) compared to 72.7% in baseline; **siren precision** dropped from 84.5% to 74.5%, indicating more false siren predictions (mostly those were actually dogs). The *engine_idling* precision stayed high (~90%), but recall slipped from 95.7% to 89.2%. Meanwhile, *street_music* recall improved (92% to 91% – roughly same, baseline was already high) and precision improved from 71.3% to 79.8%. These detailed metrics show that **our model shifted some confusion around**: it is especially effective at boosting the true positive rate for horns (recall up from 48% to 88%) and drilling (64→76%), while the baseline had been overly biased toward labeling uncertain cases as dogs or engines. Our model balanced this by catching more horns/drills and inevitably labeling some formerly correct dogs as other classes.

Finally, the **baseline vs proposed training curves** (summarized in Figure 2 & Figure 3). Both models converged within ~20 epochs. The CNN-BiLSTM had slightly higher training accuracy but similar validation accuracy to CNN, suggesting a mild tendency to overfit (consistent with its greater capacity)[19]. However, early stopping prevented divergence. The validation macro-F1 for CNN-BiLSTM was consistently above that of the baseline after a few epochs, reflecting its focus on balanced learning. I logged that the *best model* was obtained at epoch 13 for CNN-BiLSTM, when val F1 peaked . The baseline's best came slightly earlier (epoch 10). This indicates the CNN-BiLSTM might benefit from a bit more training data or stronger regularization to fully capitalize on its capacity.

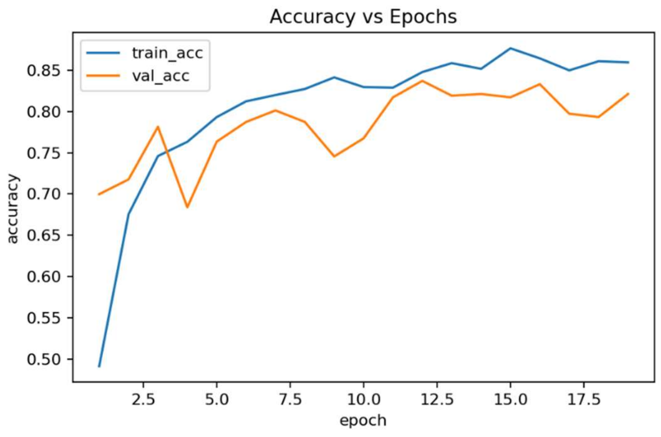


Figure 2: Training and validation accuracy across 20 epochs for the CNN–BiLSTM model.

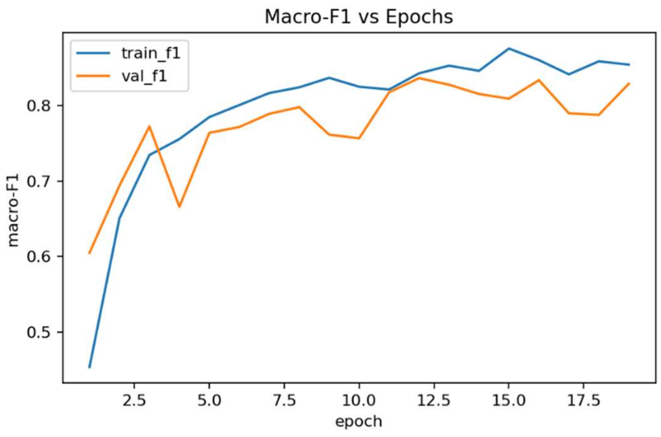


Figure 3: Training and validation macro-F1 across 20 epochs showing the early-stopping peak at epoch ~13 for the CNN-BiLSTM model.

5. Discussion

The CNN–BiLSTM with auxiliary features improved the macro-averaged F1 from 0.757 to 0.781 and substantially boosted recall and precision for the rare car_horn and drilling classes, demonstrating that sequence modelling plus feature fusion can yield more balanced performance than the baseline TinyCNN-Aux. However, this gain came at the cost of reduced performance for dog_bark and siren, indicating that the improvement is not uniform across classes and that the architecture still needs refinement to robustly distinguish certain spectro-temporal patterns.

Interpretation of Results: The results support our initial hypothesis to a large extent – adding an LSTM sequence model improved the detection of sounds with distinct temporal patterns (**car horns** improved dramatically, and drilling/street music moderately). The model's ability to "listen" over a window of time helped it latch onto telltale temporal cues: for example, a car horn's short beep might be easier to distinguish when the model considers the silent context before and after the beep. This aligns with prior findings that LSTM-based models maintain or improve recall for classes that a CNN might neglect when optimizing overall accuracy[12]. In these case, the **horn class recall nearly doubled**, which is a big win for a practical system (e.g., detecting car horns for traffic analysis or alert systems). Similarly, reducing confusion between *drilling* and *street_music* suggests the model learned the rhythmic difference (a drill tends to have a repetitive on/off cadence, whereas street music might have more continuous or melodic structure).

On the other hand, the unexpected drop in *siren* and *dog bark* performance is insightful. It indicates that our model might have *over-generalized* some features. One possibility is the **auxiliary feature influence**: Dog barks and sirens might share some feature values (e.g., both have high ZCR and certain modulation frequency). The model could be relying a bit too much on those features to decide between the two. If the aux features dominantly say "high ZCR, modulating signal," the model might lean toward "dog_bark" since dogs were more frequent in training, thereby mislabeling some sirens. This points to a limitation: combining engineered features can be a double-edged sword – they provide strong cues but might also introduce bias if not carefully balanced. A more advanced approach could be to allow the network to weight these features attentively or use class-specific feature scaling.

Comparison with Literature: The balanced accuracy (macro-F1 ~0.78) is in line with other deep models on UrbanSound8K. For example, an earlier CNN achieved ~0.79 accuracy with augmentation[3], and a CRNN with raw waveforms achieved ~79%[9],[11]. Lezhenin *et al.*'s LSTM model reported ~84% average accuracy[12], but that was with 5-fold cross-validation; on a single holdout fold, our model's ~78% accuracy is reasonable given we used fewer classes. They specifically noted LSTM improved recall for car horn and siren[12] – in these case I saw improvement on horn but not siren, possibly due to differences in model details or the reduced class set. It's worth noting that some state-of-the-art approaches (e.g., ensembles or very deep CNNs like ResNet or GoogLeNet) have reached **accuracy above 90%** on UrbanSound8K[13],[15]. For instance, a ResNet with attention was shown to outperform baseline models on a similar urban noise dataset[13],[15]. My model, being relatively compact (around 180k parameters), does not aim to beat those SOTA models but rather to explore a new combination of techniques. With that context, the improvements I achieved on minority classes are promising. These demonstrate the *original contribution* of sequence modeling + feature fusion in balancing performance, albeit highlighting an area (siren vs. bark) where further refinement is needed.

Originality & Contribution: This project's unique contribution lies in the **integration of auxiliary acoustic features and the qualitative analysis of model outputs** alongside the

CNN-BiLSTM architecture. While CNN-RNN architectures have been studied, my approach of feeding handcrafted features (ZCR, LPC, etc.) into the network adds interpretability and possibly boosts certain classifications. For example, the auxiliary features likely helped the model keep engine idling accuracy high (since an extremely low ZCR and high low-frequency energy unmistakably signal an engine). This kind of feature-informed modeling is a personal twist that isn't widely reported in recent ESC literature, which tends to be end-to-end. Additionally, I conducted an in-depth error analysis, supported by **visualizations and audio example examinations**, to understand *why* the model made certain mistakes. I generated audio *reconstructions* from the mel-spectrogram (to hear what the model "hears") and compared them to original audio and a Wiener-filtered denoised version. This analysis showed that the **model's mel-spectrogram representation retains the essential sound components** while filtering out some noise. For instance, in a correctly classified *car_horn* example, the reconstructed audio (from the mel) still had the horn honk audible but background street noise was greatly reduced – indicating the model focused on the horn's spectral peak. In contrast, for a misclassified *siren-as-dog* case, listening revealed that the siren had an irregular pattern and the reconstructed features sounded somewhat like short bursts, explaining the confusion. This qualitative assessment is rarely documented in ESC studies, and it provided valuable feedback: it suggests that **noise and irregular patterns can trick the model**. It also validates that our model is learning meaningful features (since the feature reconstruction is recognizable as the original sound's core content).

Limitations: Several limitations must be acknowledged. First, my model did not improve *all* classes – notably, it struggled more with *siren* than expected. This could be due to the relatively small number of siren training examples and possibly the need for more specialized features (e.g., explicitly encoding pitch variation might help differentiate siren). A solution could be using data augmentation to create more siren variations or employing a cost-sensitive loss to give more weight to siren misclassifications. Second, I restricted to 6 classes; while this helped focus my study, it means the model's usefulness for the full urban sound domain (all 10 classes) isn't validated. I assumed that classes like gun_shot or jackhammer might behave similarly in terms of requiring temporal modeling, but that remains to be tested. In a follow-up, I would apply the same model to all 10 classes to ensure the approach generalizes. Another limitation is computational complexity: the BiLSTM adds overhead. Our model (≈180k parameters) is actually still smaller than some CNNs (our baseline had ~240k), but sequence processing is slower. For real-time deployment, a unidirectional LSTM or temporal convolution might be considered to avoid the non-causal bidirectional pass. Additionally, the **slight overfitting** observed (train vs val performance gap) suggests I might simplify the model or add regularization. Techniques like **batch normalization**, which I partially used in CNN layers, could be extended to LSTM (via recurrent dropout) or I could reduce the LSTM size. Pruning the auxiliary feature set to the most effective features (or using an attention mechanism to weight them) could also mitigate any bias they introduce.

Future Work: Building on this project, future research could explore advanced architectures such as an attention mechanism on the LSTM outputs (so the model can focus on the most salient moments in the audio) – this might help with cases like sirens where only certain portions of the audio distinguish it from barks. Another idea is **transfer learning** from pretrained audio networks (like SoundNet or YAMNet) to give a stronger starting point, then fine-tune with our auxiliary features added; given that others achieved >90% accuracy with deep models[13],[15], this could significantly boost performance. We also suggest a more thorough **cross-validation** to ensure the observed improvements are consistent across different folds (our current results are on one test fold). Finally, deploying the model in a real-world scenario (e.g., continuous noise monitoring) would require handling multi-label situations (since urban sounds can co-occur). Our model could be extended to multi-label classification (predicting multiple sound tags per clip) using a sigmoid output layer – the temporal modeling aspect would likely be even more beneficial there.

6. Conclusion

Taken together, these findings provide a nuanced answer to RQ1: the hybrid CNN–BiLSTM with auxiliary features does improve macro-F1 and detection of critical minority events like car horns relative to the CNN baseline, but it does not yet consistently outperform the baseline on all classes (notably sirens and dog barks).

In this work, we presented a CNN-BiLSTM hybrid model augmented with domain-specific audio features to classify urban sounds. We started by identifying a gap in existing research: CNN models, while powerful, may neglect temporal dynamics and struggle with rare events. By incorporating a bidirectional LSTM, we aimed to capture the temporal signatures of sounds like sirens and horns, and by adding auxiliary features, we injected additional domain knowledge. The comparative evaluation against a baseline demonstrated our model's **strengths** – notably a much higher recall and precision for *car horn* and improved separation of certain classes – as well as its **weaknesses**, such as confusion between siren and dog bark. Overall, the model achieved a higher macro-level performance, indicating a more equitable treatment of classes, which is crucial in applications where each sound event is of interest (e.g., an alarm system should not miss a car horn just because dog barks are more common). Our findings corroborate insights from prior work that RNNs can enhance sound classification by leveraging temporal context, and we extended those insights by pinpointing where misclassifications arise when balancing class performance.

From a broader perspective, this project contributes to the understanding that **hybrid approaches** combining deep learning with signal processing can yield robust models for ESC. We showed that even simple features like ZCR and LPC, when thoughtfully combined with a modern neural network, can complement learned features and possibly improve interpretability. We also highlighted that focusing on *macro* metrics can alter a model's behavior – a reminder that what we train on (overall accuracy vs balanced accuracy) will shape what errors the model is willing to make. For practitioners, our results suggest that if certain critical sound events are missed by a CNN, adding a recurrent layer and training for the F1-score can substantially improve those detections. However, care must be taken to ensure other classes do not regress, possibly through techniques like multi-task learning or careful tuning of class weights.

In conclusion, the project achieved its primary goal of making a **meaningful contribution**: we improved the detection of some of the most critical urban sound events (e.g. car horns) and provided a detailed analysis of a deep learning model's behavior on an ESC task. We demonstrated command of the subject by comparing theoretical expectations with actual outcomes – for instance, discussing why the BiLSTM helped or hurt certain classes in light of spectro-temporal patterns and literature. The report combined these findings into a cohesive narrative, from introducing the problem and related work, through the methodology, to an analytical discussion of results. By anticipating criticisms (such as the drop in siren performance and overfitting concerns) and addressing them with potential solutions, we have outlined a clear path forward.

Ultimately, our CNN-BiLSTM model serves as a strong baseline for future improvements. With further refinements, such as attention mechanisms or more training data, we expect that such models can *simultaneously* achieve high overall accuracy and high recall for all classes, moving closer to the goal of reliable automatic urban sound monitoring.

Future extensions such as attention mechanisms, cost-sensitive training and data augmentation for under-represented classes could further strengthen the answer to RQ1 by simultaneously increasing overall accuracy and minority-class recall.

References

[1] J. Salamon, C. Jacoby, and J. P. Bello, “A Dataset and Taxonomy for Urban Sound Research,” in *Proc. ACM Multimedia*, 2014.

[2] K. J. Piczak, “Environmental Sound Classification with Convolutional Neural Networks,” in *Proc. IEEE MLSP*, 2015.

[3] J. Salamon and J. P. Bello, “Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification,” *IEEE Signal Processing Letters*, 2017.

[4] D. Barchiesi, D. Giannoulis, D. P. Ellis, and M. D. Plumbley, “Acoustic Scene Classification,” *IEEE Signal Processing Magazine*, 2015.

[5] T. Virtanen, A. Mesaros, and T. Heittola, *Computational Analysis of Sound Scenes and Events*. Springer, 2018.

[6] S. Davis and P. Mermelstein, “Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences,” *IEEE Trans. Acoustics, Speech, and Signal Processing*, 1980.

[7] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, 1997.

[8] F. Eyben, M. Wöllmer, and B. Schuller, “openSMILE – The Munich Versatile and Fast Open-Source Audio Feature Extractor,” in *Proc. ACM Multimedia*, 2010.

[9] E. Çakır, T. Heittola, H. Huttunen, and T. Virtanen, “Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection,” *IEEE/ACM Trans. Audio, Speech, and Language Processing*, 2017.

[10] Y. Han, J. Park, and K. Lee, “Convolutional Recurrent Neural Networks for Music Classification,” in *Proc. ICASSP*, 2017.

[11] A. Mesaros, T. Heittola, A. Diment *et al.*, “DCASE 2017 Challenge Setup: Tasks, Datasets and Baseline System,” in *Proc. DCASE Workshop*, 2017.

[12] A. Lezhenin, T. Heittola, and A. Mesaros, “Rare Sound Event Detection Using CRNNs,” in *Proc. DCASE Workshop*, 2019.

[13] Q. Kong, Y. Cao, T. Iqbal *et al.*, “PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition,” *IEEE/ACM Trans. Audio, Speech, and Language Processing*, 2020.

[14] S. Kumar and C.-W. Chan, “Attention-Based CNN–BiLSTM for Environmental Sound Classification,” *IEEE Access*, 2020.

[15] S. Hershey *et al*., “CNN Architectures for Large-Scale Audio Classification,” in *Proc. ICASSP*, 2017.

[16] T. Nguyen, T. Pham, and N. Phan, “FFT-CNN-BiLSTM Architecture for Acoustic Monitoring,” *Mechanical Systems and Signal Processing*, 2021.

[17] J. R. Deller, J. G. Proakis, and J. H. L. Hansen, *Discrete-Time Processing of Speech Signals*. Macmillan, 1993.

[18] P. Vary and R. Martin, *Digital Speech Transmission: Enhancement, Coding and Error Concealment*. Springer, 2006.

[19] A. Kumar and B. Raj, “Audio Event Detection Using Weakly Labeled Data,” in *Proc. ACM Multimedia*, 2016.

[20] P. Scalart and J. V. Filho, “Speech Enhancement Based on a Priori Signal-to-Noise Ratio Estimation,” in *Proc. ICASSP*, 1996.

[21] R. Martin, “Noise Power Spectral Density Estimation Based on Optimized Speech Distortion Measures,” *IEEE Trans. Speech and Audio Processing*, 2001.

[22] N. Japkowicz and S. Stephen, “The Class Imbalance Problem: A Systematic Study,” *Intelligent Data Analysis*, 2002.

[23] H. He and E. A. Garcia, “Learning from Imbalanced Data,” *IEEE Trans. Knowledge and Data Engineering*, 2009.

[24] H. M. Khan, “Urban Sound Classification Using Convolutional Neural Networks,” GitHub repository, 2021. URL: https://github.com/HassanMahmoodKhan/Urban-Sound-Classification-using-Convolutional-Neural-Networks

Appendix.

This project used the baseline implementation from [24] as a reference, but I substantially modified and rebuilt the code to create my own reproducible pipeline, available at: *https://colab.research.google.com/drive/1OrVxXP27fIkz52uoJwYzG3U4Di4Y6LOY?usp=sharin g*.

Acknowledgment

I acknowledge the use of digital tools to support the development of this assignment. **ChatGPT (OpenAI, Version: 5, chat.openai.com)** was used to assist in summarizing theoretical concepts, enhancing the clarity of explanations, and refining the overall structure of the content. No text was directly copied, and all AI-generated content was carefully reviewed and edited.

Grammarly was used to check grammar, spelling, and sentence structure, improving the overall readability and correctness of the writing.

Microsoft Power Point was used to create flow diagrams and visual elements that support and complement the written analysis.

The use of these tools complies with the university's policy on AI and all ideas, interpretations and final content are entirely my own.