Audio Engineering Society

Conference Paper 21

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

From CNN to Reservoir Computing: A New Perspective on Acoustic Scene Classification

Yuxuan He[1], Alireza Molla Ali Hosseini[1], Jakob Abeßer[2], Lina Jaurigue[1], Alexander Raake[1], and Kathy Lüdge[1]

[1]*Technische Universität Ilmenau, Ilmenau, Germany*
[2]*Fraunhofer IDMT, Ilmenau, Germany*

Correspondence should be addressed to Yuxuan He (yuxuan.he@tu-ilmenau.de)

ABSTRACT

Acoustic Scene Classification (ASC) is a fundamental task in machine listening, aiming to categorize the acoustic environment in which an audio recording was made. Although Convolutional Neural Networks (CNNs) are still the dominant approach in ASC today, increasing attention has been given to optimizing model efficiency, particularly for model deployment in resource-limited computational platforms such as hearing aids. In this study, we explore Reservoir Computing (RC) as an alternative lightweight approach for ASC and systematically compare its performance against multiple CNN architectures, including ResNet-34, MobileNet-V2, EfficientNet-B2, and TF-SepNet-20. Our results indicate that CNN models consistently outperform RC in classification accuracy, even when trained without data augmentation. However, RC exhibits significantly lower computational cost, requires less training time, and achieves the fastest inference speed among all models. This trade-off between accuracy and efficiency suggests that model selection should be guided by application requirements: CNNs are preferable when accuracy is paramount, while RC provides an alternative for low-power real-time applications such as hearing aids.

1 Introduction

Hearing loss is a growing global health concern, affecting more than 11% of the population in the European Union, with projections that more than 2.5 billion people worldwide will suffer from hearing impairment by 2050 [1]. Despite the availability of hearing aids, only 37-41% of individuals with hearing difficulties use these devices, mainly due to persistent issues with speech comprehension in noisy environments and the complexity of manual adjustment processes [2, 3].

Addressing these challenges, modern hearing aid technology incorporates adaptive signal processing systems to enhance speech perception in complex acoustic environments [4]. A key objective is to enable hearing aids to dynamically adjust to individual hearing profiles of users and daily auditory scenes, improve speech intelligibility in noisy conditions, and adapt to diverse listening environments [3]. A fundamental functionality in achieving this adaptability is Acoustic Scene Classification (ASC), which involves recognizing and categorizing environmental soundscapes into predefined classes, such as shopping malls, metro stations, or urban streets [5]. Incorporating ASC into hearing aids allows real-time adaptation to changing acoustic scenes, improving speech intelligibility and user comfort. For example, an ASC module can detect a

transition from a quiet office to a noisy street and automatically adjust the noise suppression and microphone directivity accordingly [3, 6].

However, the development of ASC models for hearing aids introduces a critical trade-off between accuracy and computational efficiency. Hearing aids operate under strict resource constraints, requiring models that achieve robust scene recognition while minimizing memory usage and power consumption [7, 8]. Therefore, the design and optimization of learning architectures play a pivotal role in determining the overall performance of ASC systems. Convolutional Neural Networks (CNNs) have proven to be highly effective in developing ASC systems [9]. Given the increasing demand for efficient audio classification models, recent research has focused on exploiting the powerful feature extraction capabilities of CNNs to improve both the accuracy and computational efficiency of ASC systems [7]. However, while CNNs remain the most popular choice, recent research suggests that Reservoir Computing (RC) offers a lightweight alternative [10], especially suitable for neuromorphic sensor integration.

RC is an efficient framework for processing time-dependent data, utilizing the transient dynamics of a fixed, nonlinear recurrent network (known as the reservoir). Near bifurcation points, the inherent nonlinearity of the system enables rich transformations of input signals into high-dimensional representation. Only the output layer is trained, significantly reducing training time and computational cost compared to traditional recurrent neural networks [10, 11]. The concept was developed to overcome the challenges of training complex neural networks using a fixed and randomly connected reservoir that captures rich temporal dynamics with minimal resource use [12, 13]. Here in this work, we want to exploit the dynamic response of the sensor for ASC in a RC setup as an energy efficient alternative to the neural networks models, leveraging the temporal correlations naturally present in audio signals.

In ASC, the task is to identify the environment from an audio recording, which requires capturing both temporal and non-linear sound characteristics. RC is well suited for this purpose because it naturally processes sequences and highlights dynamic features such as amplitude changes, frequency shifts, and temporal patterns, improving the distinction between scenes such as busy streets, parks, and bustling airports [14]. Moreover, recent advances allow the reservoir to be implemented with physical devices such as lasers or memristors. These hardware reservoirs can perform the high-dimensional projection mechanically, further reducing computational costs. Optimizing hyperparameters directly on such devices offers a promising path toward faster, more energy-efficient, and potentially more accurate audio scene classification systems [15, 16, 17, 18, 19].

Selecting the optimal learning architecture is crucial for achieving accurate and efficient ASC performance in hearing aids. This study makes the following contributions: (1) we introduce a novel RC model for ASC, offering an alternative to conventional deep learning approaches; (2) we systematically compare the RC model with four CNN architectures—ResNet-34, EfficientNet-B2, MobileNet-V2, and TF-SepNet-20—evaluating their performance; (3) we provide a detailed analysis of the trade-offs between classification accuracy, computational complexity, and efficiency, highlighting how different architectures align with the constraints of real-world hearing aid applications.

2 Methods

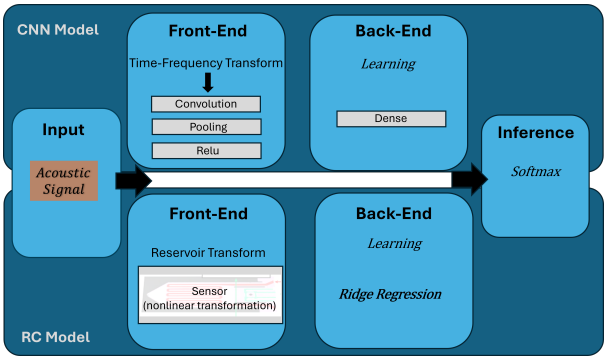


Fig. 1: Overview of the ASC processing pipeline.

To systematically compare CNN-based and RC-based approaches for ASC, we adopt a structured framework that decomposes each model into two key components: a front-end for feature extraction and a back-end for classification, as illustrated in Fig. 1.

In the front-end of an ASC model, raw audio input is transformed into a structured representation suitable for machine learning. In the case of common CNN-based models, this process involves applying the Short-Time Fourier Transform (STFT) followed by a Mel-scaled

filter bank transformation and a logarithmic magnitude compression. Such Mel spectrograms are then processed by a series of convolutional layers, pooling operations, and non-linear activations, progressively extracting relevant acoustic features [20, 21]. Conversely, in the RC-based approach, the front-end consists of a reservoir transformation where the audio waveform is fed into a nonlinear dynamic system (the sensor). A Fast Fourier Transform (FFT) is then applied to extract frequency-domain features, forming the representation used for classification.

The back-end of an ASC system is responsible for learning a mapping from the extracted features to predefined scene categories. For CNNs, this stage consists of fully-connected dense layers followed by a Softmax classifier, which assigns probabilities to different scene classes [6]. In contrast, the RC model employs ridge regression as a lightweight classification method, using the transformed reservoir responses for the final scene classification.

The following sections provide a detailed description of both CNN-based and RC-based ASC systems. Section 2.1 outlines the architecture and processing pipeline of the CNN approach, while Section 2.2 details the implementation of RC for ASC.

2.1 Convolutional Neural Network

Convolutional Layers. The convolutional layer is a core building block of CNNs. The convolution operation can be expressed mathematically as:

$$S_{i,j} = (\mathbf{K} * \mathbf{X})_{i,j} = \sum_{u=1}^{k} \sum_{v=1}^{k} K_{u,v} \cdot X_{i+u,j+v} \qquad (1)$$

where $\mathbf{K} \in R^{k \times k}$ is the convolutional kernel, and i, j denote the spatial position in the output feature map.

Several important parameters govern the behavior of the convolution operation, shaping the characteristics of the resulting feature maps. One such parameter is the **stride** (s_k), which determines the step size at which the kernel moves across the input [22]. Another parameter is **zero-padding** (p_k), which involves symmetrically adding rows and columns of zeros around the input to control the output size [23]. Additionally, the **dilation rate** (d_k) significantly influences the receptive field by introducing gaps between kernel elements [24].

Receptive Field. The receptive field of a neuron in a CNN refers to the region in the input space that contributes to its activation [25]. It determines the context in which a neuron can "see" at a specific layer.

Pooling Layers. Pooling layers downsample the feature maps, reducing their spatial dimensions while retaining essential information [26]. Two common pooling techniques are **Max Pooling** and **Average Pooling**. Max pooling retains the maximum value within each pooling window, emphasizing the most salient features [27]. Average pooling computes the average value within the pooling window, smoothing the feature map:

$$S_{i,j} = \frac{1}{k^2} \sum_{u=1}^{k} \sum_{v=1}^{k} X_{i+u,j+v} \qquad (2)$$

2.1.1 Feature Extraction for CNN

To effectively process audio inputs for our CNN models, as illustrated in Figure 1, we first apply a preprocessing pipeline to extract time-frequency representations. The input audio is provided in a single-channel format with a sampling rate of 44.1 kHz. Specifically, the STFT is performed using a window size of 3072 samples (64 ms) and a hop size of 500 samples (10.42 ms). Log-Mel spectrograms are computed using a Mel-scaled filter bank with 256 frequency bands and 4096 FFT points. Each spectrogram is stored as a three-dimensional tensor $\mathbf{X} \in R^{256 \times 84 \times 1}$, capturing 256 frequency bins, 84 time frames, and one channel which is processed by the CNN model.

2.1.2 CNN Architectures

As a performance comparison for the RC model in ASC, we evaluated four commonly used CNN architectures: ResNet-34 [28], EfficientNet-B2 [29], MobileNet-V2 [30], and TF-SepNet-20 [31].

ResNet-34 ResNet [28] is a family of deep CNNs designed to overcome the vanishing gradient problem through residual connections, which facilitate stable training in deep architectures. Among its variants, ResNet-34 adopts a bottleneck-free residual block design, preserving spatial and temporal details while maintaining a moderate parameter size. In this study, ResNet-34 is selected for its strong feature extraction capabilities and effective hierarchical modeling of spectro-temporal patterns, making it a well-suited candidate for ASC tasks [6, 32].

MobileNetV2 MobileNet [33] is a family of lightweight CNNs designed for efficiency, particularly in resource-constrained environments. It employs depthwise separable convolutions to reduce computational complexity, while maintaining strong feature extraction capabilities. Among its variants, MobileNetV2 [30] introduces inverted residual blocks, which enhance representational power with minimal computational cost. In this study, we adopt MobileNetV2 for its balance between accuracy and efficiency [34, 35].

EfficientNet-B2 EfficientNet [29] is a family of CNNs designed for optimal model scaling through compound scaling, which balances depth, width, and resolution to improve accuracy while minimizing computational costs. Among its variants, EfficientNet-B2 offers an trade-off between efficiency and performance, making it particularly suitable for ASC tasks that require spectro-temporal pattern recognition with low computational overhead. In this study, EfficientNet-B2 is selected for its lightweight architecture, swish activation for improved gradient flow, and mobile inverted bottleneck convolution (MBConv) layers for enhanced efficiency [35, 36].

TF-SepNet-20 TFSepNet [31] is an efficient CNN architecture designed to enhance feature extraction while minimizing computational cost. Unlike conventional CNNs, it employs time-frequency separable convolution (TF-SepConv), where 1D convolutional kernels process spectral and temporal features independently before merging them, optimizing ASC-related feature extraction. The architecture is inspired by BC-ResNet [37] but replaces 2D convolutions with 1D kernels along time and frequency axes, reducing parameter counts and MACs consumption. Additionally, depthwise separable convolutions and adaptive residual normalization improve efficiency and generalization. The global average pooling layer ensures a compact final representation while maintaining classification performance [6, 31].

2.1.3 Training Setup

The training process involves optimizing the model for 100 epochs using the Adam optimizer with default hyperparameters within the PyTorch framework. The cross-entropy loss function is used, as it is standard for multi-class classification tasks. A batch size of 32 is used. To facilitate faster convergence and to improve training stability, a warm-up learning rate strategy is adopted [38]. Specifically, the learning rate is gradually increased from 0 to 0.01 over the initial five epochs. After the warm-up phase, the learning rate is progressively reduced to 0 following a cosine annealing schedule [39]. Furthermore, early stopping is used to prevent overfitting, using 20% of the training data as validation data, with a patience setting of 15 epochs. Training is stopped if validation performance does not improve for 15 consecutive epochs [40]. Moreover, it is important to note that the CNN architectures in this study were trained without data augmentation.

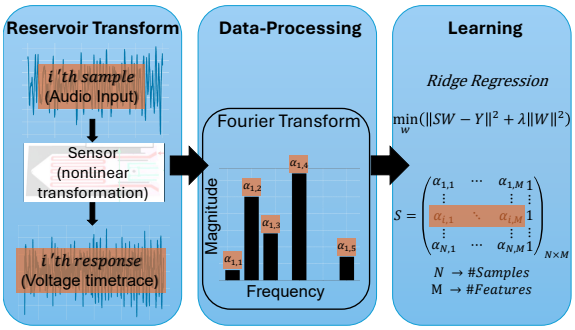


Fig. 2: The overall pipeline for the sensor based RC method using a MEMS-sensor for ASC.

2.2 Reservoir Computing Model Design

The overall pipeline of the RC method is shown in Figure 2. In the following we explain each component in detail.

Sensor-Reservoir Transform. We simulate the dynamics of a microelectromechanical (MEMS) sensor using a physics-based model derived from the Euler-Bernoulli beam theory, as detailed in [15]. This analytical model serves as the reservoir in our approach, transforming input signals through the nonlinear, auditory-like dynamic responce of the MEMS device. The MEMS sensor is described by the following set of equations:

$$\ddot{x}(t) + \frac{\omega_0}{Q_0}\dot{x}(t) + \omega_0^2 x(t) = \alpha\,\theta(t) + F_{\text{ext}}(t) \tag{3}$$

$$\dot{\theta}(t) + \beta\,\theta(t) = \frac{\gamma \cdot \min((a\,u_{\text{ac}}(t) + u_{\text{dc}})^2, u_{\text{max}}^2)}{R^2} \tag{4}$$

$$\dot{u}_{\text{ac}}(t) = -\frac{u_{\text{ac}}(t)}{\tau} + k\dot{x}(t) \tag{5}$$

Here $x(t)$, $\theta(t)$, and $u_{ac}(t)$ are beam deflection, temperature change and actuation voltage and describe the time dependent oscillation state of the sensor. The sensor is designed to mimic the behavior of the cochlea by converting external forces (i.e., sounds modelled by F_{ext}) into voltage time series $u_{ac}(t)$ using piezoresistive elements and an in-time feedback loop (feedback strenth a in Eq.(4)) for improved adaptation to its surroundings. The parameters $\alpha, \omega_0, Q_0, \gamma, \beta, \tau, k, R, u_{dc}$, $u_{\max}$ are conversion factor (deflection $x(t)$ to sensing voltages), angular resonance frequency, quality factor, conversion factor (actuation voltage u_{ac} to temperature θ), time constant (temperature changes), time constant for high pass filter, calibration factor (deflection $x(t)$ to sensing voltage), heater resistance, bias voltage, and cutoff voltage, respectively, and chosen to model the actual device from [15].

We implement the parameters for an existing sensor [15] and derive guideline for experimental implementation within a RC setup. We focus on the effect of the feedback strength a and the dc voltage u_{dc}.

To fully understand the sensor's behavior, its dynamics are first analyzed in the absence of external forces. This analysis includes investigating resonance frequencies, bifurcations, and overall sensor responses, which are crucial for effective hyperparameter optimization. It has already been shown that the sensor dynamics exhibits a Hopf bifurcation in (u_{dc},a)-parameter space, beyond which self sustained oscillations occur[41]. In Fig.3a, the steady state amplitude x of the sensor response is color coded with grey while the number of maxima observed in x in the self oscillatory regimes is depicted with blue/pink colors. The Hopf bifurcation is visualized via white and yellow dashed lines as determined numerically from our simulations. The yellow and white dashed lines indicate the dynamic transitions from steady state to the self-oscillatory behavior in the low-frequency and high-frequency regimes, respectively, resulting from a Hopf bifurcation. Exemplary time series of the sensor deflection $x(t)$ are shown in Fig.3b,c.

Comparing positive and negative feedback strength, we find that the sensor exhibits two distinct dynamic regimes which are: 1) A regime showing high resonance frequency near 14 kHz for negative a (timeseries in Fig. 3c), and 2) A regime with a complex dynamic response consisting of low frequencies (from 40 Hz to 250 Hz) and the 14 kHz component (timeseries in Fig. 3b).

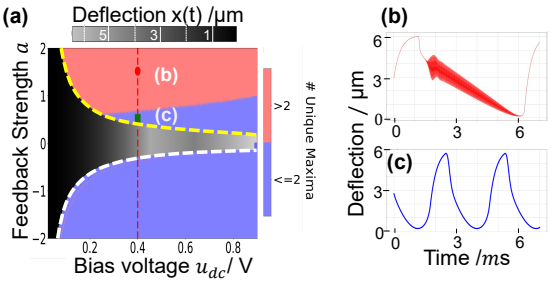


Fig. 3: Sensor dynamics (a) Unique maxima found in the sensor deflection $x(t)$ color coded in blue/red as a function of bias voltage u_{dc} and feedback strength a. Grey colors depict deflection value for the case without oscillations. White and yellow dashed lines represent the Hopf-bifurcation. (b) Time series of $x(t)$ with both high and low frequency components at $u_{\mathrm{dc}} = 0.4$, $a = 0.52$, (c) $x(t)$ with only low-frequency components at $u_{\mathrm{dc}} = 0.4$, $a = 1.52$.

Simulation and Data-Processing. We simulate the RC performance of the sensor using an audio scene dataset composed of one-second mono recordings at a 44,100 Hz sample rate. These recordings serve as the external force $F_{ext}(t)$ that acts on the sensor. The dynamic response of the sensor is given by the ODE system presented in Eqs.(3)-(5) which we simulate using a fourth-order Runge-Kutta method with a time step of 10^{-6} which is the largest step size that fully captures the system's dynamics without loss of accuracy. The audio samples are first interpolated, passed through a low-pass filter to eliminate aliasing effects, and then fed into the ODE system in order to obtain the sensor response u_{ac}. Subsequently we post-process the response u_{ac} by applying a Fast Fourier Transform (FFT) to time series where each has a length of 1×10^6. Each sequence i of a Fourier transformed time series is one sample (number of samples is N). The FFT is given by

$$X(k) = \sum_{n=0}^{N'-1} x(n)\, e^{-j\frac{2\pi kn}{N'}}, \quad k = 0, 1, \ldots, N'-1, \quad (6)$$

where $N' = 10^6$ is the length of each time series and n is the time index (the time between n and n+1 ,which is one microsecond). We then compute the magnitude $|X(k)|$ via $|X|^2 = (\mathrm{Re}X)^2 + (\mathrm{Im}X)^2$ for each frequency k, where each k represents one feature of the sensor

response. The total number of features M will be subject to change as we can select different numbers of relevant features for classification.

For the classification task, we select M frequencies (features) that are most informative for distinguishing between classes. The M magnitudes $|X(k)|$ (denoted as α_i for sample i in Fig.2) form the feature vector for each sample. The resulting state matrix S has M columns (features) and N rows (samples).

Learning via Ridge Regression. Once the state matrix S is filled with the simulated and Fourier transformed sensor responses, we normalize the obtained data so that each feature (each column in S) has a zero mean and unit variance. We then append an additional column of ones to the state matrix S to serve as a bias term[42]. We use RC for the training and thus we will need 10 ridge regression with Tikhonov regularization (parameter λ) to classify the data into ten distinct classes. The trained weights that determine the output of the reservoir computer are called weight matrix W and are determined from solving the optimization problem $\min_W \|Y - SW\|_2^2 + \lambda \|W\|_2^2$ with Y being the target vector. The closed-form solution is given by

$$\hat{W} = \left(S^T S + \lambda I\right)^{-1} S^T Y. \tag{7}$$

a winner-takes-all strategy is then applied to determine the final predicted class for each sample. Ridge regression is chosen for its computational simplicity and efficiency, as it avoids backpropagation and iterative training, making it suitable for fast and low-power applications.

Hyperparameter Optimization. In the RC approach, two key groups of hyperparameters are optimized: the sensor parameters which are bias voltage u_{dc} and feedback strength a, and the model parameters which are ridge regression regularization parameter λ and the number of features M. In order to find the optimal sensor parameters u_{dc} and a, we perform a grid search. Fig. 4 shows the performance values achieved when scanning the u_{dc}-a-parameter plane. Within the steady state regions (grey colors in Fig.3) we find the best performance (red colors in Fig.4) . Also, slight improvements are seen when the operation point is closer to the transition point from steady state to low-frequency regime (yellow line in Fig.4).

Then, with the optimized sensor parameters, we optimized the model parameters. We first tune λ to maximize the classification accuracy using ridge regression

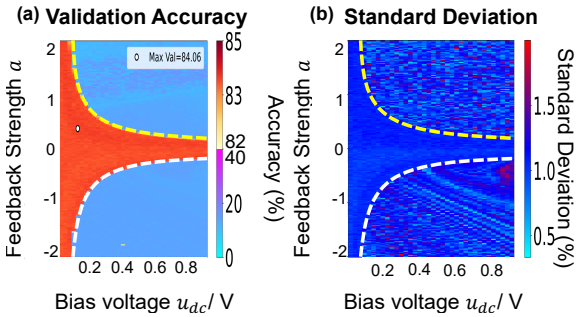


Fig. 4: Sensor Parameters (a) Average validation accuracy (from a 10-fold cross-validation) of the RC model as a function of sensor settings—bias voltage u_{dc} and feedback strength a. The white and yellow dashed lines mark the Hopf bifurcation, and the model uses $M = 24000$ features. (b) The corresponding standard deviation of these accuracies; note that the high-accuracy (red) region in (a) has a standard deviation of 1.

on the full feature set. Then, for the selected λ, the top M features are selected by analyzing the trained weights per class: weights are split into positive and negative groups, their medians are computed, and features exceeding a scaled median threshold are retained. This targeted pruning isolates the most influential features, yielding a compact representation without compromising accuracy.

3 Evaluation and Result

3.1 Dataset

In this work, we use the TAU Urban Acoustic Scenes 2022 Mobile development dataset [14], which is a well-established benchmark to assess the performance of low-complexity ASC systems. The dataset contains audio recordings from four different devices. However, for our experiments, we exclusively use the recordings from device A. This device captures the highest-quality audio using a Soundman OKM II Klassik/studio A3 binaural microphone and a Zoom F8 recorder. The decision to limit our analysis to a single device was made to minimize variability caused by different recording setups, ensuring that any observed performance gains are driven solely by the proposed model architectures. This approach allows us to isolate the impact of the model design on classification accuracy without introducing

domain shifts or device-specific effects. We follow the official 7:3 training/test split for our experiments.

3.2 Experimental Design

To systematically evaluate the performance of the RC model in comparison with CNN architectures under different data conditions, we design two experiments. The first experiment investigates model performance under limited data diversity by restricting the training set to a single city, specifically Barcelona, from the dataset. The training data is split into 80% for training and 20% for validation, and the same split is applied across all models to ensure a fair comparison. This setting allows us to assess the performance of different models when data is scarce. The second experiment examines the effect of increasing data diversity. We expand the training data to include recordings from three and five different cities, enabling us to analyze how increasing the number of cities in the training data impacts the performance of the RC model compared to CNN architectures. Both experiments are conducted using the four CNN models ResNet-34, EfficientNet-B2, MobileNetV2, and TF-SepNet, as described in Section 2.1.2. This setup allows us to explore the relationship between model size and classification ability across different data conditions and to better understand whether the RC model is particularly advantageous when training data is limited.

3.3 Computational Complexity

When comparing the computational complexity of RC and CNN models, it is important to acknowledge that the methods for calculating the number of parameters and Multiply-Accumulate operations (MACs) differ significantly due to their distinct architectural structures.

Parameter and MACs Calculation in CNNs. In CNN, the total parameter count is obtained by summing up the parameters from all convolutional and fully connected layers [43]. The total MACs is defined as the sum of preprocessing MACs and CNN MACs. The preprocessing MACs account for the operations required to transform raw audio waveforms into log-Mel spectrograms, which serve as inputs to the CNN. Moreover, to facilitate the calculation of MACs in our CNN models, we employed the open-source tool *NeSsi* [43].

Parameter and MACs Calculation in RC. In the ridge regression classifier for a 10-class problem in RC for this work, the state matrix S is composed of N rows (samples) and M columns (features). To account for the bias term, an additional column of ones is appended to S, resulting in an effective feature dimension of $M+1$. Consequently, the weight matrix W that maps the augmented feature space to the 10 output classes has dimensions $(M+1)\times 10$. Therefore, the total number of parameters in the model is given by $(M+1)\times 10$. The total MACs in RC is also defined as the sum of data-processing MACs and RC model MACs. The data-processing MACs account for the operations required to apply the real FFT on sensor responses [44]. Moreover, in order to have similar measure as the CNN models, the calculation of MACs in RC model also employed the open-source tool *NeSsi* [43].

3.4 Results

The results of **Experiment 1** are summarized in Table 1. Regarding validation accuracy, all three CNN architectures exhibit consistently high performance, with values exceeding 95%. In contrast, the RC model shows a lower validation accuracy of 84.49% using the optimal configuration ($u_{dc}=0.1, a=-1.08, \lambda=1e4, M=1260$). In terms of test accuracy, the CNN models also demonstrate stable performance, with all three basic architectures achieving results above 60%. TF-SepNet-20, while having a lower test accuracy than the other CNNs, still achieves 56.00%. The RC model, on the other hand, reaches a test accuracy of 44.50%. Regarding model complexity, both TF-SepNet-20 (0.016M) and the RC model (0.012M) exhibit significantly lower parameter counts compared to the other three CNN architectures. In terms of inference time (IT), among the CNN-based models, ResNet-34 achieves the fastest inference, requiring only 3.93 ms. This is faster than MobileNet-V2 (7.15 ms) and EfficientNet-B2 (10.87 ms). However, the most efficient model overall is RC, with an IT of only 1.70 ms, highlighting its suitability for resource-constrained scenarios like hearing aids.

The results of **Experiment 2**, illustrated in Figure 5. For validation accuracy, the CNN models show consistently high performance, maintaining values around 90% or higher under different training data conditions. In contrast, the RC model shows a noticeable decline in validation accuracy as the diversity of training data increases. Starting from 84.49% with 1 city, its validation accuracy drops to 72.38% with 3 cities and further decreases to 66.06% with 5 cities. In terms of test accuracy, all four CNN models remain relatively stable,

Architecture	Params (M)	MACs (G)	TT (s)	IT (ms)	Valid. Acc. (%)	Test Acc. (%)
ResNet-34	21.283	29.81	266.44	3.93	95.39	62.75
EfficientNet-B2	9.214	5.82	993.13	10.87	98.10	64.75
MobileNet-V2	2.236	2.54	446.67	7.15	97.19	64.50
TF-SepNet-20	0.016	0.27	557.89	4.95	93.96	56.00
Reservoir Computing	0.012	0.01	0.21	1.70	84.49	44.50

Table 1: Experiment 1: Model Performance on a Single City Dataset. The table presents a comparative analysis of different architectures in terms of model complexity (Params, MACs), computational efficiency, and classification accuracy (Validation Accuracy and Test Accuracy). TT refers to Training Time, and IT refers to Inference Time.

fluctuating around 60%, with minor variations across different training conditions. The RC model also exhibits relatively stable test accuracy, maintaining values around 45%. However, unlike its validation accuracy, the RC test accuracy shows a slight upward trend as the diversity of training data increases, improving from 44.50% with 1 city to 49.75% with 5 cities.

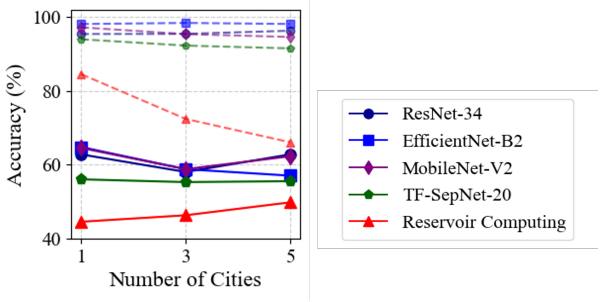


Fig. 5: Experiment 2: Impact of Increasing Data Diversity on Model Performance. Test accuracy (%) and validation accuracy (%) trends across different city data splits. Solid lines represent test accuracy, while dashed lines indicate validation accuracy.

4 Conclusion

The experimental results highlight the trade-off between classification performance and computational efficiency in ASC models. The CNN models consistently outperformed RC in classification accuracy, with all four CNN variants achieving test accuracy around 60%, while the RC model maintaining test accuracy around 45–50%. However, the RC model demonstrated remarkable efficiency in both TT and IT. Unlike CNNs, which require extensive training, the RC model required less than 1s for training across all experimental settings. Its IT also remained lower than that of CNNs, making it an attractive option for real-time applications with strict computational constraints. Such low inference latency, combined with its lightweight nature, makes the RC model particularly suitable for deployment in hearing aids, where energy efficiency and real-time responsiveness are critical.

Rather than presenting a best model, our findings suggest two viable solutions depending on system requirements: CNNs for accuracy-driven applications and RC for efficiency-critical scenarios. This flexibility can be beneficial, as it allows system designers to tailor ASC implementations according to specific needs. The ability to select between a high-accuracy model and an ultra-efficient model provides a pathway for optimizing ASC deployment in practical applications, particularly for hearing assistive technologies where balancing accuracy and computational cost is crucial.

This work presents an initial and straightforward step towards developing an RC model for ASC. While our results are promising, there is significant potential for further improvement. Moreover, adapting RC to other tasks may require careful reconfiguration of the sensor setup, as the reservoir behavior is tied to the physical sensor's dynamics. Therefore, task-specific tuning and modeling are essential. Future research may focus on optimizing the sensor setup and incorporating multiple sensors in an optimal configuration. Such advancements could lead to enhanced performance, potentially allowing the system to operate directly on the raw reservoir time series responses without the need for extensive intermediate data processing.

References

[1] W. H. Organization, "World report on hearing," *World Report on Hearing*, 2021.

[2] S. Kochkin, "Marketrak vii: Obstacles to adult non-user adoption of hearing aids," *The Hearing Journal*, vol. 60, p. 24–51, 2007.

[3] R. Vanitha Devi and Vasundhara, "Review on recent advances in hearing aids: A signal processing perspective," in *Proceedings of the International Conference on Paradigms of Computing, Communication and Data Sciences* (R. P. Yadav, S. J. Nanda, P. S. Rana, and M. H. Lim, eds.), Algorithms for Intelligent Systems, Springer, Singapore, 2023.

[4] C. Lenk, K. Ved, S. Durstewitz, T. Ivanov, M. Ziegler, and P. Hövel, "Bio-inspired, neuromorphic acoustic sensing," in *Bio-Inspired Information Pathways* (M. Ziegler, T. Mussenbrock, and H. Kohlstedt, eds.), vol. 16 of *Springer Series on Bio- and Neurosystems*, Springer, Cham, 2024.

[5] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, pp. 16–34, May 2015.

[6] B. Ding, T. Zhang, C. Wang, G. Liu, J. Liang, R. Hu, Y. Wu, and D. Guo, "Acoustic scene classification: A comprehensive survey," *Expert Systems with Applications*, vol. 238, p. 121902, 2024.

[7] Y. Li, W. Cao, W. Xie, Q. Huang, W. Pang, and Q. He, "Low-complexity acoustic scene classification using data augmentation and lightweight resnet," in *2022 16th IEEE International Conference on Signal Processing (ICSP)*, vol. 1, pp. 41–45, October 2022.

[8] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[9] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, September 2015.

[10] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural. Comput.*, vol. 14, p. 2531, Nov. 2002.

[11] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," GMD Report 148, GMD - German National Research Institute for Computer Science, 2001.

[12] L. Larger, A. Baylón-Fuentes, R. Martinenghi, V. S. Udaltsov, Y. K. Chembo, and M. Jacquot, "High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification," *Phys. Rev. X*, vol. 7, p. 011015, February 2017.

[13] M. Nakajima, K. Tanaka, and T. Hashimoto, "Scalable reservoir computing on coherent linear photonic processor," *Commun. Phys.*, vol. 4, no. 20, p. 2399, 2021.

[14] T. Heittola, A. Mesaros, and T. Virtanen, "Tau urban acoustic scenes 2022 mobile, evaluation dataset," June 2022.

[15] C. Lenk, P. Hövel, K. Ved, S. Durstewitz, T. Meurer, T. Fritsch, A. Männchen, J. Küller, D. Beer, T. Ivanov, and M. Ziegler, "Neuromorphic acoustic sensing using an adaptive microelectromechanical cochlea with integrated feedback," *Nat. Electron.*, vol. 6, pp. 370–380, May 2023.

[16] P. Antonik, F. Duport, M. Hermans, A. Smerieri, M. Haelterman, and S. Massar, "Online training of an opto-electronic reservoir computer applied to real-time channel equalization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, p. 2686, August 2016.

[17] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutierrez, L. Pesquera, C. R. Mirasso, and I. Fischer, "Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing," *Opt. Express*, vol. 20, no. 3, pp. 3241–3249, 2012.

[18] K. Vandoorne, P. Mechet, T. van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nat. Commun.*, vol. 5, p. 3541, Mar. 2014.

[19] B. Barazani, G. Dion, J.-F. Morissette, L. Beaudoin, and J. Sylvestre, "Microfabricated neuroaccelerometer: Integrating sensing and reservoir computing in mems," *J. Microelectromech. Syst.*, vol. 29, no. 3, pp. 338–347, 2020.

[20] V. K. Singh, K. Sharma, and S. N. Sur, "A survey on preprocessing and classification techniques for acoustic scene," *Expert Systems with Applications*, vol. 229, p. 120520, November 2023.

[21] J. Abesser, "A review of deep learning based methods for acoustic scene classification," *Applied Sciences*, vol. 10, p. 2020, March 2020.

[22] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[23] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, May 2018.

[24] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015.

[25] W. Luo, Y. Li, R. Urtasun, and R. S. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," *ArXiv*, vol. abs/1701.04128, 2016.

[26] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013.

[27] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), vol. 8689 of *Lecture Notes in Computer Science*, pp. 818–833, Springer, Cham, 2014.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

[29] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *ArXiv*, vol. abs/1905.11946, 2019.

[30] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[31] Y. Cai, P. Zhang, and S. Li, "Tf-sepnet: An efficient 1d kernel design in cnns for low-complexity acoustic scene classification," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 821–825, 2024.

[32] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 131–135, 2017.

[33] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *ArXiv*, vol. abs/1704.04861, 2017.

[34] A. G. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1314–1324, 2019.

[35] X. Zhang, W. Xie, and Y. Gong, "Lightweight acoustic scene classification using knowledge distillation and efficient cnns," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 2055–2067, 2022.

[36] I. J. Ananya, S. Suad, S. H. Choudhury, and M. A. Khan, "A comparative study on approaches to acoustic scene classification using cnns," *ArXiv*, vol. abs/2204.12177, 2022.

[37] B. Kim, S. Chang, J. Lee, and D. Sung, "Broadcasted residual learning for efficient keyword spotting," in *Interspeech*, 2021.

[38] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *ArXiv*, vol. abs/1706.02677, 2017.

[39] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv: Learning*, 2016.

[40] L. Prechelt, "Early stopping-but when?," in *Neural Networks*, 1996.

[41] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Boca Raton: CRC Press, 2nd edition ed., 2015. eBook Published 23 May 2019.

[42] T. Hülser, F. Köster, L. C. Jaurigue, and K. Lüdge, "Role of delay-times in delay-based photonic reservoir computing," *Opt. Mater. Express*, vol. 12, no. 3, p. 1214, 2022.

[43] A. Ancilotto, "Nessi: Neural network simple statistics interface." `https://github.com/AlbertoAncilotto/NeSsi`, 2023. Accessed: 2025-01-31.

[44] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.