# Exploring Performance–Complexity Trade-Offs in Sound Event Detection Models

Tobias Morocutti[2], Florian Schmid[1], Jonathan Greif[1], Francesco Foscarin[1,2], Gerhard Widmer[1,2]

[1]*Institute of Computational Perception*, [2]*LIT Artificial Intelligence Lab*

*Johannes Kepler University*, Linz, Austria

{tobias.morocutti, florian.schmid}@jku.at

*Abstract*—We target the problem of developing new low-complexity networks for the sound event detection task. Our goal is to meticulously analyze the performance-complexity trade-off, aiming to be competitive with the large state-of-the-art models, at a fraction of the computational requirements. We find that low-complexity convolutional models previously proposed for audio tagging can be effectively adapted for event detection (which requires frame-wise prediction) by adjusting convolutional strides, removing the global pooling, and, importantly, adding a sequence model before the (now frame-wise) classification heads. Systematic experiments reveal that the best choice for the sequence model type depends on which complexity metric is most important for the given application. We also investigate the impact of enhanced training strategies such as knowledge distillation. In the end, we show that combined with an optimized training strategy, we can reach event detection performance comparable to state-of-the-art transformers while requiring only around $5\%$ of the parameters. We release all our pre-trained models and the code for reproducing this work to support future research in low-complexity sound event detection[1].

*Index Terms*—Sound Event Detection, Low-Complexity, CNNs, Sequence Models

## I. INTRODUCTION

In recent years, the pursuit of better performance has led to ever larger deep learning models. The field of sound event detection (SED), i.e., the identification and exact temporal localization of specific sound events in audio files, has followed this trend, with state-of-the-art models reaching 90M parameters [1]–[5]. However, larger models incur significant inference costs, which can be prohibitive for applications with limited computational resources, such as embedded systems.

Prior research on low-complexity audio classification has leveraged efficient convolutional neural network (CNN) architectures adapted from the vision domain [6]–[8] or introduced custom-designed efficient CNNs [9]–[13]. However, most of these models are optimized for clip-wise predictions, i.e., predicting a class label for an entire given audio clip. In contrast, the goal in SED is to identify specific acoustic events in a recording, including begin and end times, which implies a need for frame-wise predictions [14].

Unlike clip-wise tasks, where local features from the CNN are typically aggregated using global pooling, SED commonly employs a sequence model on top of the CNN to capture long-term temporal dependencies. Efficient SED architectures frequently utilize depthwise separable convolutions to reduce complexity [15]–[17], yet the optimal choice of sequence model remains an open question. While recurrent neural networks (RNNs) are a widely adopted default—forming the basis of the popular convolutional recurrent neural network (CRNN) architecture [18]–[21]—recent research has also shown promise in lightweight attention-based models [15], [17].

The present paper addresses this gap by systematically comparing various sequence models built upon a fixed CNN backbone, aiming to identify sequence models that offer the best trade-off between performance and complexity. We choose MobileNetV3 [22] as the CNN backbone, as it is one of the most widely used depthwise separable CNNs and has demonstrated strong performance in large-scale audio tagging [8]. By fixing the CNN architecture, we focus our analysis entirely on the temporal modeling stage.

To get the most meaningful results, we evaluate on the largest publicly-available SED dataset - *AudioSet Strong* [23] - and use the state-of-the-art training pipeline [4], which includes knowledge distillation [24] from large transformer models. Our experiments aim to answer the following key questions:

- Which sequence model, when varying its size, yields the best performance-complexity trade-off? (Section IV-A)
- Do the benefits of using a sequence model hold across multiple sizes of the CNN backbone? (Section IV-B)
- How important are training techniques, such as knowledge distillation, for low-complexity SED? (Sections IV-C and IV-D)

## II. ARCHITECTURES

We assume the input of our task to be 10-second audio files, encoded as mel spectrograms with 128 frequency bins and 1000 time frames.

All the architectures we propose consist of 3 main components: a frame-wise convolutional backbone $g$ (see Section II-A) that inputs the audio spectrogram $\mathbf{x}$, a sequence model $f$ (see Section II-B) that processes the output $\hat{\mathbf{z}}$ of the backbone, and the task heads that produce the frame-wise sound event predictions $\hat{\mathbf{o}}_C$.

Formally, our model can be represented as:

$$\hat{\mathbf{z}} = g(\mathbf{x})$$
$$\{\hat{\mathbf{o}}_C\}_{t=1}^{250} = \mathbf{W_h}f(\mathbf{W_d}\{\hat{\mathbf{z}}\}_{t=1}^{250} + \mathbf{b_d}) + \mathbf{b_h}$$

where $\mathbf{W_d}$ and $\mathbf{b_d}$ represent a linear projection aligning embeddings with the dimension of the sequence model $f$. If no sequence model is used, $f$ and the linear projection ($\mathbf{W_d}$, $\mathbf{b_d}$) are identity functions. The output of $f$ is then transformed by a position-wise linear layer, parameterized by $\mathbf{W_h}$ and $\mathbf{b_h}$, to generate frame-level predictions $\{\hat{\mathbf{o}}_C\}_{t=1}^{250}$, with dimension $C$ corresponding to the number of classes in the task.

## A. Frame-wise Convolutional Backbone

As the convolutional backbone, we use MobileNetV3 [22]. To enable frame-wise instead of clip-wise predictions, we remove the global pooling and the global classification heads. The outputs of this network are then three-dimensional embeddings $\hat{\mathbf{z}}$ of shape *channel × freq × time*. Compared to the original MobileNet, we increase some convolutional strides along the frequency axis from 1 to 2 while decreasing others along the time axis from 2 to 1. This changes the shape of $\hat{\mathbf{z}}$ from *channels × 4 × 32* to *channels × 1 × 250*, matching the temporal resolution of the pseudo-labels used for knowledge distillation as described in Section III-B. Therefore, each of the 250 time frames in $\hat{\mathbf{z}}$ corresponds to a 40-millisecond segment of the original 10-second audio clip. We refer to this adapted model as *frame-wise MobileNet* (*fmn*).

For the experiments that require it, we scale the frame-wise MobileNets following the method described in [8], maintaining the number of layers while adjusting the model's width (by multiplying the default number of channels of all convolutional layers) using a scaling factor $\alpha$. We consider models with $\alpha \in [0.4, 0.6, 1.0, 2.0, 3.0]$ which we name *fmn04*, *fmn06*, *fmn10*, *fmn20* and *fmn30*, respectively.

## B. Sequence Models

To investigate whether sequence models can improve the performance-complexity trade-off of low-complexity CNNs, we evaluate a range of well-established as well as recently proposed sequence models. Our experiments include transformer blocks (TF) [25], multi-head self-attention (ATT) [25], bidirectional gated recurrent units (BiGRU) [26], temporal convolutional networks (TCN) [27], Mamba blocks (MAMBA) [28], [29], and a hybrid sequence model (HYBRID) that combines a minimal version of a GRU (minGRU) [30] with a self-attention layer.

For TF and ATT, we incorporate rotary positional embeddings [31], because these performed better than other positional encodings for small hidden dimensions in our preliminary studies. The TCN blocks consist of five convolutional layers with exponentially increasing dilation values, where the $n$-th layer has a dilation of $2^n$. TCN's model capacity is defined by the number of convolutional channels, analogous to the hidden dimension in recurrent models and the model dimension in attention-based and state-space models. For simplicity, we refer to the sequence model dimension as the hidden dimension throughout this work. For MAMBA, we use the more recent Mamba-2 blocks [29] with a state dimension of 64.

HYBRID consists of two parallel branches, following the design of Hymba [32]. However, inspired by [33], we replace Mamba with a bidirectional minGRU [30] and use it in parallel with self-attention.[2] The outputs of these branches are averaged and linearly projected before serving as input to the subsequent sequence model block or the task head.

## III. EXPERIMENTAL SETUP

In this section, we outline our dataset, training configurations, performance metrics, and the methods used to assess the complexity of the proposed models.

## A. Dataset

AudioSet Strong [23] includes predefined training and evaluation sets containing 103,463 and 16,996 ten-second audio clips, respectively. Of these, we successfully downloaded 100,911 training clips and 16,935 evaluation clips. We train on all 447 available sound classes in the training set, while, in line with [1], [4], we evaluate on the 407 classes that overlap between the training and evaluation sets.

## B. Training Configuration

We initialize the backbone CNNs with ImageNet [35] pre-trained weights and train them on AudioSet Weak [36], following the knowledge distillation routine from [8]. However, we find that extending the training to 300 epochs improves performance.

Following the AudioSet Weak pre-training, the full models (pre-trained backbone + sequence model) are trained on AudioSet Strong for 120 epochs using an Adam optimizer with no weight decay, a cosine learning rate schedule with 5,000 warmup steps, and a batch size of 256.

Our preliminary experiments showed that the best performance was achieved by using different learning rates for the pre-trained backbone and the randomly initialized sequence model. Specifically, for the backbone, we use a layer-wise learning rate decay of 0.9 and choose the learning rate with independent grid searches per model size over 1e-3, 3e-3, and 6e-3 without using any sequence model. We then select the sequence model learning rate independently for every type and size from a small, pre-defined grid ({5e-4, 8e-4, 3e-3}). Following [4], we train our models using knowledge distillation on pseudo-labels generated by an ensemble of 15 transformers, in addition to the typical one-hot encoded labels. The distillation and hard label losses are weighted at 0.9 and 0.1, respectively. Furthermore, the frame-wise MobileNet is designed to align with the resolution (40 ms) of these pseudo-labels. Training our low-complexity models on the pseudo-labels enables a simple training setup without overfitting, using mixup [37] as the only data augmentation.

To ensure a fair comparison among the sequence models, we optimize the median filter window size for postprocessing on *fmn10* using no sequence model and apply the same setting

---

[2]In HYBRID, we use bidirectional minGRUs instead of BiGRUs, as they require fewer parameters and can be efficiently parallelized using the Heinsen Parallel Scan Algorithm [34]. Since our focus is on offline inference, this algorithm can be leveraged for inference, significantly increasing throughput. In contrast to the unidirectional minGRUs implemented in https://github.com/lucidrains/minGRU-pytorch, we employ bidirectional minGRUs for improved performance in our application.

across all experiments. Odd, centered window sizes yield significantly better performance than even ones, with a size of 9 frames (0.36 seconds) leading to the best results for *fmn10*.

### C. Performance and Complexity Metrics

Following [4], we assess performance using PSDS1 [38], a threshold-independent metric that quantifies the intersection between ground truth and detected events. By prioritizing low reaction time and precise event localization, PSDS1 aligns well with our primary research focus.

We use three measures of model complexity: the parameter count, the number of multiply-accumulate (MAC) operations[3], and throughput, defined as the number of samples processed per second during inference. Throughput and MACs are computed using ten-second audio clips. Throughput is measured on an Nvidia RTX 3090 GPU with a batch size of 64 and 16-bit floating-point precision, considering the PyTorch implementation of the models, which we publicly release.

## IV. RESULTS

### A. Performance-Complexity Trade-Off of Sequence Models

In this experiment, we investigate the performance-complexity trade-off of different sequence models built on top of *fmn10*. We vary their size across four values of hidden dimension (128, 256, 512, and 1024) and compare their performance, total parameter count, throughput, and MACs.

We explore stacking multiple sequence model blocks and observe that, across different sequence models, using two blocks results in more stable training and a notable performance improvement over a single block. However, adding a third block further increases the complexity without delivering a meaningful performance gain. Therefore, we consistently stack two blocks of the sequence model on top of the convolutional backbone.

Figure 1 shows that all sequence models improve the performance of *fmn10*, even with the smallest hidden dimension. However, only models incorporating TF, BiGRU, ATT, and HYBRID achieve performance comparable to or surpassing *fmn20* while maintaining lower parameter counts, lower MACs, and increased throughput, resulting in a more favorable performance-complexity trade-off. This shows that adding suitable sequence models on top of *fmn10* scales more efficiently than simply scaling up the convolutional backbone itself. In contrast, MAMBA and TCN fail to reach the performance level of *fmn20* at any hidden dimension size.

The choice of the model offering the best performance-complexity trade-off depends on the complexity metric of interest. The top choices for parameters, throughput, and MACs are HYBRID, TF, and BiGRU, respectively. For example, *fmn10* using TF and a hidden dimension of 256 (*fmn10+TF:256*) matches the performance of *fmn20* while achieving 1.76 times its throughput and requiring only 37.6% of its total parameters. Similarly, *fmn10* with HYBRID at a hidden dimension of
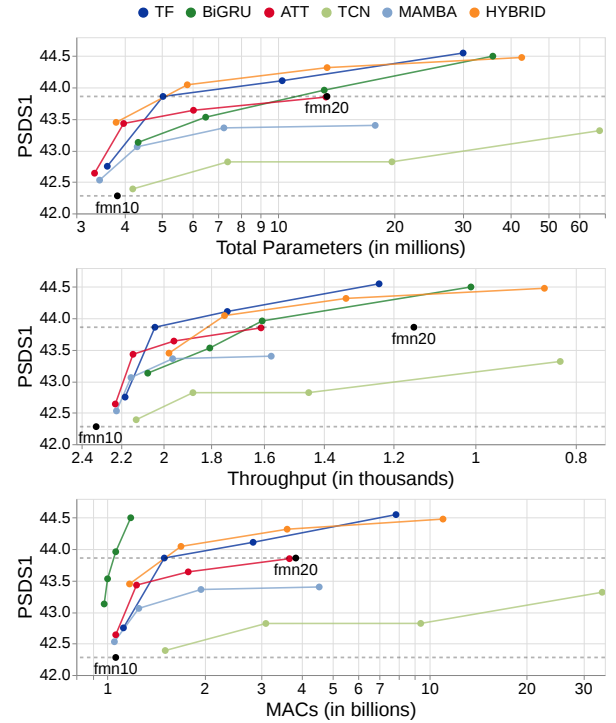


Fig. 1. **Performance-Complexity trade-off** of *fmn10* trained with various sequence models, scaled across four complexity levels. Colored points indicate increasing hidden dimensions (128, 256, 512, and 1024) from left to right. Evaluated sequence models include transformer blocks (TF), BiGRUs, self-attention layers (ATT), temporal convolutional networks (TCNs), Mamba blocks (MAMBA), and a hybrid sequence model (HYBRID) combining minGRU and self-attention. For reference, *fmn10* and *fmn20* without sequence models are shown as well.

128 (*fmn10+HYBRID:128*) has fewer parameters than *fmn10* without a sequence model[4] yet improves the PSDS1 score from 42.28 to 43.45. However, *fmn10+HYBRID:128* exhibits lower throughput compared to *fmn10+TF:256*. In terms of MACs, BiGRU is the most effective[5], but its sequential processing limits parallel computation, resulting in lower throughput compared to TF and HYBRID. We conclude that TF is the most balanced choice with strong results on all three complexity measures. In particular, *fmn10+TF:256* strikes an excellent trade-off between performance gain and minimal added complexity, making it the preferred model for further experiments.

### B. Scaling the Backbone and Sequence Model

In the previous section, we analyzed the performance-complexity trade-off of sequence models using a fixed-size convolutional backbone (*fmn10*). Here, we extend this analysis by jointly scaling both the convolutional backbone and the sequence model. As outlined in Section II, the frame-wise

---

[3]We use the model profiler contained in Microsoft's DeepSpeed framework to measure MACs.

[4]If the down-projection, sequence model, and linear head together require fewer parameters or MACs than a single large linear layer mapping embeddings to target classes, adding a small sequence model can be more efficient than using no sequence model with respect to these metrics.

[5]BiGRUs involve multiple operations, such as gating, point-wise products, and point-wise summations, which do not count as MACs.
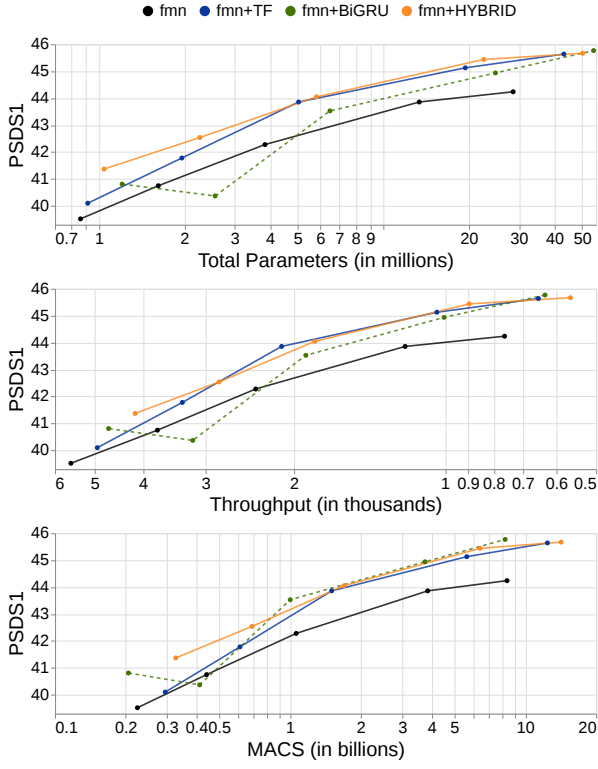
Fig. 2. **Performance-Complexity trade-off** of the models scaled to different complexity levels. The points on the black line, from left to right, correspond to the models *fmn04*, *fmn06*, *fmn10*, *fmn20*, and *fmn30*. We compare these to models incorporating transformer blocks (*fmn+TF*), HYBRID (*fmn+HYBRID*), or BiGRUs (*fmns+BiGRU*) with sequence models scaled proportionally to the backbone. For clarity, the BiGRU line is dashed.

TABLE I
IMPACT OF KD AND CNN BACKBONE FINE-TUNING.

| Model | KD | Frozen CNN Backbone | PSDS1 |
|---|---|---|---|
| *fmn10* | ✗ | ✗ | 38.11 |
| *fmn10* | ✓ | ✗ | 42.28 |
| *fmn10+TF:256* | ✗ | ✗ | 41.13 |
| *fmn10+TF:256* | ✓ | ✗ | **43.86** |
| *fmn10+TF:256* | ✓ | ✓ | 42.05 |

MobileNets are scaled by a factor $\alpha$, which we also apply to the sequence models' hidden dimension. Based on the findings in Section IV-A, we set $\alpha = 1.0$ to correspond with a hidden dimension of 256 and evaluate the three top-performing sequence models: TF, HYBRID, and BiGRU.

Three key conclusions can be drawn from Figure 2: 1) TF and HYBRID consistently outperform fmns without sequence models across all scaling factors; 2) HYBRID performs best in the lower complexity range, with *fmn04+HYBRID* and *fmn06+HYBRID* surpassing their TF and BiGRU counterparts; and 3) TF and HYBRID exhibit greater stability than BiGRU, as the performance drop of BiGRU from $\alpha = 0.4$ to $\alpha = 0.6$ is a consistent trend across multiple independent runs.

### C. Impact of KD and CNN Backbone Fine-Tuning

As outlined in Section III-B, we first pre-train the CNN backbone on AudioSet Weak and fine-tune the full model

TABLE II
IMPROVED KNOWLEDGE DISTILLATION VARIANTS.

| Model | Weak | Extended Training | PSDS1 |
|---|---|---|---|
| *fmn10+TF:256* | ✗ | ✗ | 43.86 |
| *fmn10+TF:256* | ✓ | ✗ | 44.70 |
| *fmn10+TF:256* | ✓ | ✓ | **45.25** |

(backbone + sequence model) on AudioSet Strong using KD. Table I presents the resulting performance effects of this setup. Regardless of whether a sequence model (*TF:256* on top of *fmn10*) is used, KD significantly enhances performance. Similarly, fine-tuning the convolutional backbone for frame-wise SED predictions further improves results, though its impact is smaller than that of KD.

### D. Expanding Knowledge Distillation to AudioSet Weak

This section explores techniques to further enhance the proposed low-complexity models, narrowing the gap to large state-of-the-art transformers. So far, we have followed the KD training routine from [4], which uses pre-computed transformer ensemble pseudo-labels on the AudioSet Strong training set. However, the distillation process is not limited to frame-wise annotated data (AudioSet Strong) and can be extended to other recordings. Following this idea, we leverage the top-performing transformer model on AudioSet Strong from [4], BEATs [39], to generate frame-level predictions for the AudioSet Weak training split (~2 million files). The distillation loss is then computed on batches containing 50% AudioSet Weak and 50% AudioSet Strong clips.

Table II shows that incorporating AudioSet Weak for distillation boosts the performance of *fmn10+TF:256* by nearly 2%, increasing its PSDS1 score from 43.86 to 44.70. Given the large scale of AudioSet Weak, we extend training from 120 to 240 epochs, yielding an additional 1.2% improvement and reaching a PSDS1 score of 45.25. These results highlight that an efficient model like *fmn10+TF:256*, combined with an optimized training strategy, can achieve performance comparable to state-of-the-art transformers (PSDS1 scores between 45.4 and 46.5 in [4]) while using only 5 million parameters—significantly fewer than the ~90 million in transformer models[6].

### V. CONCLUSION

In this paper, we proposed and thoroughly evaluated efficient sound event detection (SED) architectures. We utilized modified MobileNetV3 models, adjusted to various complexity levels, as the convolutional backbone, and combined them with different sequence models. Our findings demonstrate that pairing appropriate sequence models with the backbone consistently outperforms fully convolutional versions across all complexity levels tested. The optimal sequence model depends on the prioritized complexity metric. We further emphasized the critical role of knowledge distillation in achieving high performance and showed that, with an optimized

---

[6]This enhanced distillation procedure is not applied to all experiments in this paper due to substantially higher training costs.

distillation strategy, our models can match state-of-the-art transformer performance while using only around 5% of their parameters. To aid future work, we release our pre-trained models along with our training and evaluation framework, inviting researchers to develop new sequence models that push the performance–complexity trade-off even further.

## REFERENCES

[1] X. Li, N. Shao, and X. Li, "Self-supervised audio teacher-student transformer for both clip-level and frame-level tasks," *Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 1336–1351, 2024.

[2] N. Shao, X. Li, and X. Li, "Fine-tune the pretrained ATST model for sound event detection," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

[3] F. Schmid, P. Primus, T. Morocutti, J. Greif, and G. Widmer, "Multi-iteration multi-stage fine-tuning of transformers for sound event detection with heterogeneous datasets," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2024.

[4] F. Schmid, T. Morocutti, F. Foscarin, J. Schlüter, P. Primus, and G. Widmer, "Effective pre-training of audio transformers for sound event detection," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

[5] P. Cai, Y. Song, K. Li, H. Song, and I. McLoughlin, "MAT-SED: A masked audio transformer with masked-reconstruction based pre-training for sound event detection," in *Proceedings of the Interspeech Conference*, 2024.

[6] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-scale pretrained audio neural networks for audio pattern recognition," *Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[7] Y. Gong, Y. Chung, and J. R. Glass, "PSLA: improving audio tagging with pretraining, sampling, labeling, and aggregation," *Transactions on Audio, Speech, and Language Processing*, 2021.

[8] F. Schmid, K. Koutini, and G. Widmer, "Efficient large-scale audio tagging via transformer-to-cnn knowledge distillation," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

[9] S. Verbitskiy, V. B. Berikov, and V. Vyshegorodtsev, "Eranns: Efficient residual audio neural networks for audio pattern recognition," *Pattern Recognit. Lett.*, vol. 161, pp. 38–44, 2022.

[10] F. Schmid, K. Koutini, and G. Widmer, "Dynamic convolutional neural networks as efficient pre-trained audio models," *Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 2227–2241, 2024.

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

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

[13] F. Schmid, T. Morocutti, S. Masoudian, K. Koutini, and G. Widmer, "Distilling the knowledge of transformers and cnns with cp-mobile," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2023.

[14] N. Turpault, R. Serizel, J. Salamon, and A. P. Shah, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2019.

[15] T. K. Chan and C. S. Chin, "Lightweight convolutional-iconformer for sound event detection," *IEEE Trans. Artif. Intell.*, vol. 4, pp. 910–921, 2023.

[16] K. Drossos, S. I. Mimilakis, S. Gharib, Y. Li, and T. Virtanen, "Sound event detection with depthwise separable and dilated convolutions," in *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*, 2020.

[17] M. Yang, L. Peng, L. Liu, Y. Wang, Z. Zhang, Z. Yuan, and J. Zhou, "*LCSED*: A low complexity CNN based SED model for iot devices," *Neurocomputing*, vol. 485, pp. 155–165, 2022.

[18] E. Çakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *Transactions on Audio, Speech, and Language Processing*, vol. 25, pp. 1291–1303, 2017.

[19] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[20] Y. Wang, J. Li, and F. Metze, "A comparison of five multiple instance learning pooling functions for sound event detection with weak labeling," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[21] Y. Li, M. Liu, K. Drossos, and T. Virtanen, "Sound event detection via dilated convolutional recurrent neural networks," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[22] A. Howard, R. Pang, H. Adam, Q. V. Le, M. Sandler, B. Chen, W. Wang, L. Chen, M. Tan, G. Chu, V. Vasudevan, and Y. Zhu, "Searching for mobilenetv3," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[23] S. Hershey, D. P. Ellis, E. Fonseca, A. Jansen, C. Liu, R. C. Moore, and M. Plakal, "The benefit of temporally-strong labels in audio event classification," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[24] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning Workshop*, 2014.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[26] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

[27] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *Speech Synthesis Workshop*, 2016.

[28] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *CoRR*, vol. abs/2312.00752, 2023.

[29] T. Dao and A. Gu, "Transformers are ssms: Generalized models and efficient algorithms through structured state space duality," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[30] L. Feng, F. Tung, M. O. Ahmed, Y. Bengio, and H. Hajimirsadeghi, "Were rnns all we needed?" *CoRR*, vol. abs/2410.01201, 2024.

[31] J. Su, M. H. M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[32] X. Dong, Y. Fu, S. Diao, W. Byeon, Z. Chen, A. S. Mahabaleshwarkar, S. Liu, M. V. Keirsbilck, M. Chen, Y. Suhara, Y. Lin, J. Kautz, and P. Molchanov, "Hymba: A hybrid-head architecture for small language models," *CoRR*, vol. abs/2411.13676, 2024.

[33] P. Wang, "minGRU-pytorch," https://github.com/lucidrains/minGRU-pytorch, 2024, GitHub repository.

[34] F. A. Heinsen, "Efficient parallelization of an ubiquitous sequential computation," *CoRR*, vol. abs/2311.06281, 2023.

[35] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[36] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An ontology and human-labeled dataset for audio events," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[37] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proceedings of the International Conference on Learning Representations (ICLR)* , 2018.

[38] J. Ebbers, R. Haeb-Umbach, and R. Serizel, "Threshold independent evaluation of sound event detection scores," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[39] S. Chen, Y. Wu, C. Wang, S. Liu, D. Tompkins, Z. Chen, W. Che, X. Yu, and F. Wei, "BEATs: Audio pre-training with acoustic tokenizers," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.