Memory-Efficient Sound Event Detection using Multi-Head Gated Recurrent Unit

Maxim K. Surkov
ITMO University
Saint Petersburg, Russia
surkovmax007@mail.ru

***Abstract*—Sound event detection is a critical task with widespread applications in industrial and scientific domains. However, most state-of-the-art methods rely on large neural networks with a substantial number of parameters, rendering them unsuitable for deployment in resource-constrained environments. Additionally, these approaches typically employ a single recurrent neural network as a decoder to predict the probabilities of event occurrences at specific timestamps. In contrast, this paper introduces a model featuring a multi-head gated recurrent unit decoder, which achieves competitive prediction accuracy while significantly reducing the number of trainable parameters. Our proposed model attains an F_1-score of 62.53% and a polyphonic sound detection score of 0.4979 on the DESED evaluation dataset, requiring only 3.2 million trainable parameters, thereby achieving state-of-the-art performance among low-complexity models. Furthermore, we provide a comprehensive analysis of the proposed approach. First, we demonstrate that multi-heading enables the creation of models with comparable or superior prediction accuracy while utilizing significantly fewer trainable parameters. Second, we show that our approach is more effective for improving the model performance compared to simply increasing the hidden size of the network. Finally, we explore the limitations of the proposed technique in scenarios where auxiliary pretrained transformers are used to generate additional informative embeddings from input audio recordings.**

I. Introduction

The task of automatic meta-information recognition involves identifying and extracting various types of information from an audio signal. This includes data such as speech [1], unusual sounds or anomalies [2], emotions [3], specific events [4], acoustic environments [5], and other elements present in the sound. In today's world, automatic audio processing is essential for many applications. For example, smart devices with advanced audio analysis capabilities are becoming increasingly popular. At the same time, researchers are actively exploring ways to improve the recognition of different types of audio data, as seen in studies presented at the annual Detection and Classification of Acoustic Scenes and Events (DCASE) conference [6]–[14]. A key focus of this work is sound event detection (SED), which aims to identify specific sound events in an audio recording and provide labels along with their start and end timestamps.

The state-of-the-art approach for addressing the SED task involves converting the input audio signal into a log-mel spectrogram, processing it through an ensemble of large transformer-based models such as PaSST [15], BEATs [16], and ATST [17], and then decoding the resulting embeddings using a recurrent neural network (RNN). The most accurate solution for DCASE Task 4 achieves a polyphonic sound detection score (PSDS) of 0.68 and utilizes 1 billion trainable parameters [18]. Consequently, the described system is not suitable for deployment within smart devices due to its high computational complexity and substantial resource requirements.

An alternative method for solving the SED problem involves replacing the large transformer encoder with a memory-efficient convolutional neural network (CNN) combined with embeddings extracted from a frozen BEATs transformer [19]. However, this system cannot be considered low-complexity due to the computational and storage costs associated with the BEATs model. Moreover, when analyzing the system without including BEATs, it becomes evident that the crucial part of trainable parameters is concentrated in the RNN decoder. In this paper, we address the challenge of reducing the number of trainable parameters in the RNN decoder.

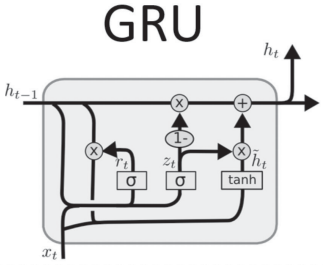


Fig. 1. Architecture of gated recurrent unit (GRU)

One of the most efficient RNN-based decoders is the gated recurrent unit (GRU) [20]. The model operates by maintaining a hidden state h_{t-1} and updating it to generate the next state h_t based on the input vector x_t. The update process is governed by a series of gating mechanisms, as illustrated in Fig. 1 and mathematically defined in Equations 1,2,3,4.

$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}) \quad (1)$$

$$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}) \quad (2)$$

$$\tilde{h}_t = \tanh\left(W_{i\tilde{h}}x_t + b_{i\tilde{h}} + r_t \odot (W_{h\tilde{h}}h_{(t-1)} + b_{h\tilde{h}})\right) \quad (3)$$

$$h_t = (1 - z_t) \odot \tilde{h}_t + z_t \odot h_{(t-1)} \quad (4)$$

where h_t is the hidden state at time t, x_t is the input at time t, $h_{(t-1)}$ is the hidden state of the layer at time $(t-1)$ or the initial hidden state at time 0, and $r_t, z_t, \tilde{h}_t$ are the reset, update, and new gates, respectively. σ is the sigmoid function, and $\odot$ is the Hadamard product.

Let D denote the dimensionality of the input vector x_t, and let H represent the dimensionality of the hidden state. Under these conditions, the GRU contains $\mathcal{O}(HD+H^2)$ trainable parameters. In this work, we propose a novel model constructed by replacing a single GRU with an ensemble of k GRUs, each with a hidden size h such that $hk = H$. This modification results in minimal degradation in model prediction accuracy while reducing the total number of trainable parameters to $\mathcal{O}(k(H/k)D + k(H/k)^2) = \mathcal{O}(HD + H^2/k)$. This approach demonstrates a significant reduction in model size while preserving performance, offering a more efficient alternative for resource-constrained applications. Moreover, the presented model achieves new state-of-the-art SED performance in terms of F-measure among low-complexity models on the DCASE Task 4 public evaluation dataset [9].

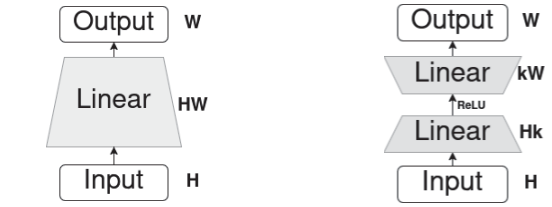
II. Method

A. Conventional method

In this section, we introduce the conventional sound event detection method. In mathematical terms, the prediction system corresponds to the operation $Y = D_{\theta_D}(E_{\theta_E}(X))$, where E denotes an encoder, D is a decoder, θ_E represents the set of trainable parameters in the encoder model, θ_D is the set of weights in the decoder model, $X = (x_1, \ldots, x_N)$, with $x_i \in \mathbf{R}^D$, is the sequence of log-mel spectrogram feature vectors, and N is the total number of time frames. Let us denote $E_{\theta_E}(X) = Q = (q_1, \ldots, q_M)$ as the sequence of embeddings obtained by the encoder model. Finally, $Y = D_{\theta_D}(Q)$ is the output of the network. Under these designations, the model complexity can be defined as the number of trainable parameters $|\theta_E| + |\theta_D|$.

The SED task involves determining the timestamps corresponding to the occurrence of each predefined sound event. In this work, we utilize a set of 10 sound events from DCASE 2024 Task 4 [9]: alarm bell ringing, blender, cat, dishes, dog, electric shaver or toothbrush, frying, running water, speech, and vacuum cleaner. A significant challenge in this task is the presence of overlapping time segments associated with different event types. To address this issue, most state-of-the-art models generate outputs in the form $Y \in \mathbf{R}^{N\times 10}$, where $Y_{t,c}$ represents the probability of the existence of sound event $c \in \{1, \ldots, 10\}$ at time frame $t \in \{1, \ldots, N\}$.

One of the most effective approaches to solving the SED task is to fine-tune an ensemble of large transformer models, such as PaSST [15], BEATs [16], and ATST [17], using the standard binary cross-entropy (BCE) loss function. The prediction pipeline is further enhanced by applying median filtering to the outputs, which improves the PSDS [21], a widely-used evaluation metric for SED. An alternative approach involves employing a low-complexity CNN to generate embeddings, concatenating these embeddings with feature vectors produced by an auxiliary pretrained BEATs model, and decoding the predictions using a recurrent neural network (RNN).



(a) Conventional linear layer (b) LoRa

Fig. 2. Architecture of the low-rank adaptation module

However, the aforementioned approaches involve the use of large neural networks, which prevents them from being considered low-complexity models. Nevertheless, there exists a series of systems that can be classified as low-complexity models [22], [23]. These systems demonstrated competitive performance until 2023, when the authors of DCASE Task 4 introduced the possibility of using large neural networks as auxiliary models to improve overall prediction accuracy, disregarding their size. Conventional low-complexity approaches are fundamentally similar to resource-intensive models, with the primary difference being the inclusion of a large pretrained encoder as an additional source of generating useful and informative embeddings from audio recordings. As a result, most compact models consist of a CNN encoder and an RNN decoder without any additional large subsystems. In this work, we consider such approaches as baseline models.

B. The proposed method

Let us examine the complexity of the GRU decoder. The model comprises three linear layers with $\mathcal{O}(HD)$ parameters and three linear layers with $\mathcal{O}(H^2)$ weights. By optimizing the utilization of these linear layers, it becomes possible to reduce the overall model complexity. The conceptual foundation of our work is inspired by the following techniques in deep learning.

The first technique is referred to as low-rank adaptation (LoRa) [24]. Initially, this method was developed for the efficient fine-tuning of large language models (LLMs). The core idea of LoRa is based on the observation that most linear layers in LLMs exhibit low-rank properties. Consequently, such layers, which typically consist of $H \times W$ weights, can be replaced with two linear layers of dimensions $H \times k$ and $k \times W$, separated by an activation function. Fig. 2 demonstrates the scheme of the explained model. This approach reduces the number of parameters in the layer from HW to $(H + W)k$. Notably, it has been demonstrated that this technique preserves the prediction accuracy of the model.

The second method involves utilizing a series of independent prediction models and aggregating their outputs to produce the final result. This collection of models is referred to as an ensemble. This technique has demonstrated state-of-the-art performance across various domains, including computer

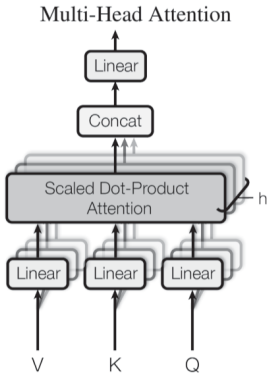
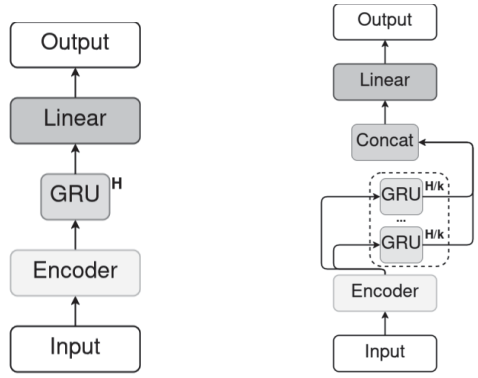


Fig. 3. Architecture of the multi-head self-attention taken from [28]



(a) Conventional GRU (b) Multi-head GRU

Fig. 4. Architecture of the proposed multi-head GRU decoder

vision (CV) [25], natural language processing (NLP) [26], and automatic speech recognition (ASR) [27].

The third approach is known as multi-head self-attention (MHSA) [28]. Conventional self-attention computes relationship scores between each pair of input vectors and generates output embeddings based on the relative importance of each input vector. To achieve this, the self-attention module projects input vectors into a single vector space of dimensionality H, where these operations are performed. In contrast, multi-head self-attention projects input vectors into k independent vector spaces, each of dimensionality H/k as shown in Fig. 3. It then performs the aforementioned operations within each subspace and concatenates the results, producing a final sequence of vectors with dimensionality $k \cdot (H/k) = H$. Multi-head attention enables the model to simultaneously attend to information from different representation subspaces at various positions. This capability is inhibited in single-head attention.

Building upon these ideas, we propose a novel multi-head recurrent neural network as a decoder model for addressing the problem of SED. Fig. 4 illustrates a schematic architecture of the invented approach. Specifically, we replace a conventional GRU with a hidden size of H with an ensemble of k smaller GRUs, each with a hidden size of H/k. We refer to these GRUs as heads and the entire model as a multi-head GRU. While the original GRU generates an output vector of dimensionality H, our proposed ensemble produces a series of vectors, each of dimensionality H/k. These vectors are concatenated into a single vector of dimensionality $k \cdot (H/k) = H$, which is then propagated through a final linear layer to obtain the output probabilities, consistent with the conventional model. By replacing one GRU with $\mathcal{O}(DH + H^2)$ parameters with k smaller GRUs, each with $\mathcal{O}(DH/k + (H/k)^2)$ parameters, the total number of trainable weights in our model is reduced to $\mathcal{O}\left(k(DH/k + (H/k)^2)\right) = \mathcal{O}(DH + H^2/k)$.

III. Experiments

A. Dataset

We utilize the DESED dataset [9], which contains 85.1 hours of 10-second recordings with labels of varying granularity: 9.6 hours of strongly-labeled samples, which include precise timestamps of event occurrences; 4.4 hours of weakly-labeled examples, which only indicate the presence of event types; 27.8 hours of synthetically generated data; and 3.3 hours of validation data. All audio samples are resampled to a 16 kHz sampling rate and subsequently transformed into log-mel spectrograms. These spectrograms are generated using 128-channel filter banks, computed over a 128 ms window with a 10 ms stride. Additionally, spectrogram masking augmentation is applied, with a maximum masking ratio of 15%.

B. Training

Given that the dataset consists of half unlabeled examples, we employ knowledge distillation from the state-of-the-art approach proposed by Schmid [18] to address this challenge. Specifically, we utilize classical soft distillation, leveraging the conventional mean squared error (MSE) between the predictions of the teacher and student models. During training, we employ a batch size of 48 samples. Optimization is performed using the AdamW optimizer with a cosine annealing learning rate schedule, where the maximum learning rate is set to 10^{-4}. The models are trained from default random initialization for 500 epochs until convergence, with the first 10% of the epochs allocated for the warmup stage. To ensure robustness and reliability, we report the average metric values across three independent experimental runs. We observe that the standard deviation across all runs remains within 1% of the relative metric value for all experimental setups. A single experimental run requires approximately 24 hours when executed on $1 \times$ NVIDIA Tesla V100 32GB.

C. Evaluation metrics

We evaluate the proposed model using two distinct groups of metrics: one for assessing prediction accuracy and the other for estimating model complexity. The first group of metrics includes the F_1-score and the PSDS [29] for evaluating the prediction accuracy of the SED task, with hyperparameters described in Table I. The second group focuses on model complexity and comprises the number of trainable parameters.

TABLE I. HYPERPARAMETERS OF THE EVALUATION METRICS

Metric	Hyperparameter name	Hyperparameter value
F_1	Collar	200 ms
PSDS	Detection Tolerance	0.7
	Ground Truth intersection	0.7
	Cost of instability across class	1
	Cost of CTs[1] on user experience	0
	Maximum False Positive rate	100

[1] Cross-triggers – the subset of false positives which match another labelled class of the multi-class system

TABLE II. ARCHITECTURE DETAILS OF CONSIDERED SINGLE AND MULTI-HEAD MODELS

Model	Size, M[1]	Heads	Channels in encoder
$M_2(1)$	2	1	$[8, 16, 32, 64, 128, 208, 208]$
$M_2(2)$	1.5	2	
$M_2(4)$	1.3	4	
$M_2(8)$	1.1	8	
$M_{3.5}(1)$	3.5	1	$[16, 32, 64, 128, 128, 274, 274]$
$M_{3.5}(2)$	2.6	2	
$M_{3.5}(4)$	2.2	4	
$M_{3.5}(8)$	1.9	8	
$M_4(1)$	5	1	$[32, 64, 128, 256, 256, 292, 292]$
$M_4(2)$	4	2	
$M_4(4)$	3.5	4	
$M_4(8)$	3.2	8	
$M_{0.5}(1)$	0.5	1	$[4, 8, 16, 32, 32, 108, 108]$
$M_{1.1}(1)$	1.1	1	$[8, 16, 32, 64, 64, 157, 157]$
$M_{2.2}(1)$	2.2	1	$[16, 32, 64, 128, 128, 210, 210]$
$M_4(1)$	4	1	$[16, 32, 64, 128, 128, 296, 296]$

[1] "M" corresponds to millions

D. Configurations

In our experiments, we utilize a convolutional neural network (CNN) comprising seven consecutive blocks as the encoder. Each block includes a convolutional layer with a kernel size of 3, followed by batch normalization, a ReLU activation function, and average pooling. The encoder processes the input log-mel spectrogram to produce a sequence of H-dimensional vectors, which are subsequently passed to the GRU decoder. To evaluate the impact of our proposed method on model size and performance, we conduct experiments with various configurations by adjusting the number of channels in the encoder and the hidden size of the decoder. This approach allows us to create a series of models with varying numbers of trainable parameters, facilitating evaluation across diverse scenarios. We develop three models with 2, 3.5, and 5 million trainable parameters, respectively. For each model, we investigate multi-head configurations with 1, 2, 4, and 8 heads. We denote the model configuration with initially W trainable weights in single-head mode, which is transformed into a model with k GRU heads as $M_W(k)$. The hidden size of the decoder model is equal to the number of channels in the last layer of the encoder model. Table II provides a comprehensive characterization of the proposed models. Furthermore, to assess the contribution of the multi-head decoder, we train single-head conventional networks of similar size and compare them with their multi-head counterparts. As baselines, we consider state-of-the-art low-complexity models from [22], [23]. To ensure a fair comparison, we evaluate our approach against their versions that do not incorporate any pretrained auxiliary models.

TABLE III. A COMPARISON OF BASELINE, SINGLE AND MULTI-HEAD MODELS

Model	Size, M[1]	F_1-score	PSDS
Baseline [22]	15	58%	0.4743
Baseline [23]	10	58.22%	0.456
$M_2(1)$	2	60.55%	0.4855
$M_2(2)$	1.5	61.14%	0.4875
$M_2(4)$	1.3	59.63%	0.4907
$M_2(8)$	1.1	58.84%	0.4759
$M_{3.5}(1)$	3.5	61.56%	0.4935
$M_{3.5}(2)$	2.6	61.33%	0.4940
$M_{3.5}(4)$	2.2	61.41%	0.4950
$M_{3.5}(8)$	1.9	60.39%	0.4984
$M_5(1)$	5	61.07%	0.4905
$M_5(2)$	4	61.66%	0.4964
$\mathbf{M_5(8)}$	**3.2**	**62.53%**	**0.4979**

[1] "M" corresponds to millions

TABLE IV. A COMPARISON OF STRAIGHTFORWARD MODEL MAGNIFICATION WITH THE PROPOSED MULTI-HEAD GRU

Model	Size, M[1]	F_1-score	PSDS
$M_{1.1}(1)$	1.1	59.34%	0.4800
$M_2(1)$	2	60.55%	0.4855
$M_{2.2}(1)$	2.2	61.16%	0.4914
$M_{3.5}(1)$	3.5	61.56%	0.4935
$M_4(1)$	4	60.54%	0.4970
$M_2(2)$	1.5	61.14%	0.4875
$M_{3.5}(4)$	2.2	61.41%	0.4950
$M_5(8)$	3.2	62.53%	0.4979
$M_5(4)$	3.5	61.72%	0.5015

[1] "M" corresponds to millions

E. Results

Based on the experimental results presented in Table III, it can be observed that replacing the conventional single-head GRU model with our proposed multi-head GRU not only preserves prediction accuracy but also enhances it while significantly reducing the model size. Specifically, for the M_2 model, replacing the single-head version with a two-head version results in an increase in the F_1-score from 60.55% to 61.14%, an improvement in the PSDS from 0.4855 to 0.4875, and a reduction in model size from 2 million parameters to 1.5 million, representing a 25% relative decrease. For the $M_{3.5}$ model, replacing one head with four heads leads to a marginal decrease in the F_1-score from 61.56% to 61.44%, while the PSDS increases from 0.4935 to 0.4950. Notably, the model size is significantly reduced from 3.5 million to 2.2 million parameters. Finally, for the M_5 model, replacing one head with eight heads results in a performance improvement: the F_1-score increases from 61.07% to 62.53%, the PSDS improves from 0.4905 to 0.4979, and the model size decreases from 5 million to 3.2 million parameters. Overall, the most accurate configuration, $M_5(8)$, outperforms the baseline model by 4.3% in absolute F_1-score and by 0.02 in PSDS while reducing the model size by up to three times.

Developing the most efficient model architecture is a critical aspect of solving complex tasks in machine learning. A

TABLE V. A COMPARISON BETWEEN A MULTI-HEAD GRU WITH A SINGLE-HEAD GRU WITH SMALLER HIDDEN SIZE

Model	Size, M[1]	F_1-score	PSDS
$M_{1.1}(1)$	1.1	60.33%	0.4825
$M_2(8)$	1.1	58.84	0.4792
$M_{1.9}(1)$	1.9	59.94%	0.4875
$M_{3.5}(8)$	1.9	60.39	0.498
$M_{3.2}(1)$	3.2	61.51%	0.4922
$M_5(8)$	3.2	62.53	0.4979

[1] "M" corresponds to millions

TABLE VI. A COMPARISON OF BASELINE, SINGLE AND MULTI-HEAD MODELS IN CASE OF USING AUXILIARY BEATS EMBEDDINGS

Model	Size, M[1]	F_1-score	PSDS
Baseline [30]	184	65.6%	0.5667
$M_5(1)$	6.3	65.43%	0.5117
$M_5(2)$	5.3	64.80%	0.5172
$M_5(4)$	4.8	63.69%	0.5092
$M_5(8)$	4.6	62.95%	0.5067

[1] "M" corresponds to millions

straightforward approach to improving performance is to increase the model size by adding more layers or expanding the dimensionality of the hidden state. However, as demonstrated in Table IV, our proposed method offers a more effective strategy for achieving higher prediction accuracy compared to simply increasing the hidden size. Specifically, when comparing models with increased hidden sizes, our approach yields superior results. For instance, instead of scaling a model from 1.1 million to 2 million parameters by increasing its hidden size, it is more advantageous to adopt the $M_2(2)$ model, as it demonstrates greater accuracy. Similarly, the $M_{3.5}(4)$ model outperforms the transition from $M_2(1)$ to $M_{2.2}(1)$, and the $M_5(8)$ model is preferable to scaling from $M_{2.2}(1)$ to $M_{3.5}(1)$. Furthermore, the $M_5(4)$ model provides better performance than increasing from $M_{3.5}(1)$ to $M_4(1)$. These results highlight the efficacy of our approach in achieving higher accuracy without relying solely on increasing model size.

To better understand the impact of replacing a single GRU with a series of smaller GRUs, we hypothesize that the considered model configurations may not be optimal in terms of hidden size. Specifically, we investigate whether a single-head GRU with a reduced hidden size could achieve better prediction accuracy compared to our proposed multi-head approach. To test this assumption, we reduce the hidden size of our single-head models by half, transforming $M_2(1)$, $M_{3.5}(1)$, and $M_5(1)$ into $M_{1.1}(1)$, $M_{1.9}(1)$, and $M_{3.2}(1)$, respectively. We then compare their prediction accuracy with multi-head GRU models of the same size. As shown in Table V, our proposed multi-head models consistently outperform single-head versions for models with 1.9 and 3.2 million trainable parameters. However, for extremely small models with 1.1 million parameters, the single-head version demonstrates slightly better performance. This suggests that the benefits of our approach become more pronounced as the model size increases.

Finally, considering that the current state-of-the-art approach for solving the SED problem involves using auxiliary BEATs embeddings, we evaluate our proposed method in this context. Specifically, we test our approach by incorporating embeddings from a pretrained BEATs model [16] while comparing model sizes without accounting for the storage and computational costs of the large transformer. We augment the models $M_5(1)$, $M_5(2)$, $M_5(4)$, and $M_5(8)$ with BEATs embeddings and compare them against the state-of-the-art approach proposed by the winners of DCASE 2023 Task 4 [30]. The experimental results, presented in Table VI, reveal that the impact of splitting a single-head GRU into a multi-head GRU is more pronounced when using additional BEATs embeddings. Increasing the number of heads by a factor of two leads to a significant decrease in the F_1-score by 1% in absolute terms and a reduction in the PSDS by 0.002. This degradation is likely due to the mismatch between the BEATs embedding size of 768 and the significantly smaller hidden size of the GRU, which diminishes the influence of the GRU embeddings on the overall model performance. Nevertheless, our proposed approach demonstrates competitive performance compared to baseline models, particularly considering that the authors of [30] employ an extensive ensemble of 46 models and utilize additional external datasets.

IV. LIMITATIONS

Although our proposed approach demonstrates improvements in preserving model quality while significantly reducing its size, our work is limited by the scope of encoder models considered. Specifically, we focus solely on convolutional neural networks (CNNs) as encoder models, while other powerful architectures, such as transformers, Conformer [31], and LSTM-based encoders, remain unexplored. Additionally, we restrict our investigation to GRU-based decoders, leaving open questions about the applicability of our approach to other decoder architectures, such as LSTMs, which exhibit slightly different structural properties. Furthermore, the primary goal of this work is to develop memory-efficient models, and thus we do not explore the impact of our approach on larger models, which remains an essential direction for future research. Another limitation arises from our reliance on knowledge distillation in the training pipeline, which leaves uncertainty about the effectiveness of our technique in supervised, semi-supervised, or unsupervised learning scenarios. Finally, our multi-head recurrent neural network employs a series of small GRUs with equal hidden sizes. However, it is hypothesized that greater diversity in the ensemble, such as using smaller networks with varying hidden sizes, could lead to improved performance. Investigating this possibility represents a critical area for future research.

V. CONCLUSIONS

In this work, we investigate a novel architecture designed to optimize memory consumption for solving the problem of sound event detection. Our approach involves replacing

a single GRU decoder with an ensemble of smaller GRUs, which significantly reduces memory usage while maintaining model prediction accuracy. The proposed model achieves state-of-the-art performance, with an F_1-score of 62.53% and a polyphonic sound detection score (PSDS) of 0.4979, outperforming other low-complexity models. The most effective configuration based on a single-head model with 5 million trainable parameters, where the single GRU is replaced with 8 smaller GRUs of equal hidden size. Additionally, we conduct a series of extensive experiments to evaluate the impact of the proposed approach. Our findings demonstrate that multi-head models achieve competitive performance while significantly reducing memory consumption up to three times. Moreover, we show that multi-heading is a more effective strategy for improving model quality compared to simply increasing the hidden size. We also analyze the impact of our approach when using auxiliary BEATs embeddings and conclude that multi-heading in this scenario is detrimental rather than beneficial. Finally, we discuss the limitations of our work and outline potential directions for future research.

VI. Acknowledgments

I extend my deepest gratitude to my beloved bride, Daria Tarasova, for her invaluable contributions to the manuscript review process and insightful discussions.

References

[1] M. Malik, M. K. Malik, K. Mehmood, and I. Makhdoom, "Automatic speech recognition: a survey," *Multimedia Tools and Applications*, vol. 80, pp. 9411–9457, 2021.

[2] Z. Mnasri, S. Rovetta, and F. Masulli, "Anomalous sound event detection: A survey of machine learning based methods and applications," *Multimedia Tools and Applications*, vol. 81, no. 4, pp. 5537–5586, 2022.

[3] M. S. Fahad, A. Ranjan, J. Yadav, and A. Deepak, "A survey of speech emotion recognition in natural environment," *Digital signal processing*, vol. 110, p. 102951, 2021.

[4] T. K. Chan and C. S. Chin, "A comprehensive review of polyphonic sound event detection," *IEEE Access*, vol. 8, pp. 103 339–103 373, 2020.

[5] 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.

[6] F. Schmid, T. Morocutti, S. Masoudian, K. Koutini, and G. Widmer, "Distilling the knowledge of transformers and CNNs with CP-mobile," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2023 Workshop (DCASE2023)*, 2023, pp. 161–165.

[7] N. Harada, D. Niizumi, D. Takeuchi, Y. Ohishi, and M. Yasuda, "First-shot anomaly detection for machine condition monitoring: A domain generalization baseline," *Proceedings of 31st European Signal Processing Conference (EUSIPCO)*, pp. 191–195, 2023.

[8] A. Politis, K. Shimada, P. Sudarsanam, S. Adavanne, D. Krause, Y. Koyama, N. Takahashi, S. Takahashi, Y. Mitsufuji, and T. Virtanen, "STARSS22: A dataset of spatial recordings of real scenes with spatiotemporal annotations of sound events," in *Proceedings of the 8th Detection and Classification of Acoustic Scenes and Events 2022 Workshop (DCASE2022)*, Nancy, France, November 2022, pp. 125–129. [Online]. Available: https://dcase.community/workshop2022/proceedings

[10] T. Pellegrini, I. Khalfaoui-Hassani, E. Labbé, and T. Masquelier, "Adapting a ConvNeXt Model to Audio Classification on AudioSet," in *Proc. INTERSPEECH 2023*, 2023, pp. 4169–4173. [Online]. Available: https://www.isca-speech.org/archive/pdfs/interspeech_2023/pellegrini23_interspeech.pdf

[9] N. Turpault, R. Serizel, A. Parag Shah, and J. Salamon, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," in *Workshop on Detection and Classification of Acoustic Scenes and Events*, New York City, United States, October 2019. [Online]. Available: https://hal.inria.fr/hal-02160855

[11] J. Lee, M. Tailleur, L. M. Heller, K. Choi, M. Lagrange, B. McFee, K. Imoto, and Y. Okamoto, "Challenge on sound scene synthesis: Evaluating text-to-audio generation," in *Audio Imagination: NeurIPS 2024 Workshop AI-Driven Speech, Music, and Sound Generation*, 2024.

[12] 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," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 2880–2894, 2020.

[13] X. Liu, H. Liu, Q. Kong, X. Mei, J. Zhao, Q. Huang, M. D. Plumbley, and W. Wang, "Separate what you describe: Language-queried audio source separation," in *Proc. INTERSPEECH 2022*, 2022.

[14] S. Damiano, L. Bondi, S. Ghaffarzadegan, A. Guntoro, and T. van Waterschoot, "Can synthetic data boost the training of deep acoustic vehicle counting networks?" in *Proceedings of the 2024 International Conference on Acoustics, Speech and Signal Processing (ICASSP) (accepted)*, Seoul, South Korea, April 2024.

[15] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient training of audio transformers with patchout," 2022.

[16] 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 40th International Conference on Machine Learning*, 2023, pp. 5178–5193.

[17] X. Li and X. Li, "Atst: Audio representation learning with teacher-student transformer," 2022.

[18] F. Schmid, P. Primus, T. Morocutti, J. Greif, and G. Widmer, "Improving audio spectrogram transformers for sound event detection through multi-stage training," DCASE2024 Challenge, Tech. Rep., May 2024.

[19] Y. Xiao, H. Yin, J. Bai, and R. K. Das, "Fmsg-jless submission for dcase 2024 task4 on sound event detection with heterogeneous training dataset and potentially missing labels."

[20] M. Zöhrer and F. Pernkopf, "Gated recurrent networks applied to acoustic scene classification and acoustic event detection."

[21] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

[22] K. He, X. Shu, S. Jia, and Y. He, "Semi-supervised sound event detection system for dcase 2022 task 4," DCASE2022 Challenge, Tech. Rep., June 2022.

[23] J. W. Kim, G. W. Lee, H. K. Kim, Y. S. Seo, and I. H. Song, "Semi-supervised learning-based sound event detection using frequency-channel-wise selective kernel for dcase challenge 2022 task 4," DCASE2022 Challenge, Tech. Rep., June 2022.

[24] E. J. Hu, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models," in *International Conference on Learning Representations*.

[25] Á. Casado-García and J. Heras, "Ensemble methods for object detection," in *ECAI 2020*. IOS Press, 2020, pp. 2688–2695.

[26] M. Anand, K. B. Sahay, M. A. Ahmed, D. Sultan, R. R. Chandan, and B. Singh, "Deep learning and natural language processing in computation for offensive language detection in online social networks by feature selection and ensemble classification techniques," *Theoretical Computer Science*, vol. 943, pp. 203–218, 2023.

[27] L. Deng and J. Platt, "Ensemble deep learning for speech recognition," in *Proc. interspeech*, 2014.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[29] J. Ebbers, R. Haeb-Umbach, and R. Serizel, "Threshold independent evaluation of sound event detection scores," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 1021–1025.

[30] J. W. Kim, S. W. Son, Y. Song, . Kim, Hong Kook1, I. H. Song, and J. E. Lim, "Semi-supervised learning-based sound event detection using frequency dynamic convolution with large kernel attention for DCASE challenge 2023 task 4," DCASE2023 Challenge, Tech. Rep., June 2023.

[31] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *Interspeech 2020*, 2020, pp. 5036–5040.