



DyConvMixer: Dynamic Convolution Mixer Architecture for Open-Vocabulary Keyword Spotting

Waseem Gharbieh^{1*}, Jinmiao Huang^{1*}, Qianhui Wan^{1*}, Han Suk Shim², Chul Lee²

¹LG Electronics Toronto AI Lab

²LG Electronics Artificial Intelligence Lab

waseem.gharbieh@lge.com, jinmiao.huang@lge.com, qianhui.wan@lge.com, hansuk.shim@lge.com, clee.lee@lge.com

Abstract

User-defined keyword spotting research has been gaining popularity in recent years. An open-vocabulary keyword spotting system with high accuracy and low power consumption remains a challenging problem. In this paper, we propose the DyConvMixer model for tackling the problem. By leveraging dynamic convolution alongside a convolutional equivalent of the MLP-Mixer architecture, we obtain an efficient and effective model that has less than 200K parameters and uses less than 11M MACs. Despite the fact that our model is less than half the size of state-of-the-art RNN and CNN models, it shows competitive results on the publicly available Hey-Snips and Hey-Snapdragon datasets. In addition, we discuss the importance of designing an effective evaluation system and detail our evaluation pipeline for comparison with future work.

Index Terms: Dynamic Convolution, Open-vocabulary Keyword Spotting, User-defined Keyword Spotting, Query-by-Example, ConvMixer

1. Introduction

A keyword spotting system is the key starting point of voice interactions between humans and smart devices. These devices constantly listen to their environment and only get triggered by certain keywords. The triggering words are usually predefined and users do not have the freedom to change them. However, the ability to register a user-preferred keyword will enable more personalized user interactions. The main challenge of such systems is that the keywords can be out of the training distribution and the system should also be small and efficient enough to run on an edge device with low latency.

Many open-vocabulary keyword spotting systems rely on the Query-by-Example (QbyE) approach to tackle the problem. The approach uses a distance function to compare the embeddings between the enrollment and query keywords. If the distance is smaller than a certain threshold, the query is considered to be the same keyword as the enrollment, thus triggering the system. A QbyE model usually has an encoder to convert the input audio signal to an embedding in vector space and a decoder (last few layers) to maximize the distance between different class embeddings while minimizing the distances of those that belong to the same class. Most of the QbyE systems in literature use Recurrent Neural Networks (RNNs) as the encoder to extract the embeddings [1, 2, 3, 4] since RNNs can project variable-duration inputs onto a fixed-size vector representation. Convolutional Neural Networks (CNN) based models, on the other hand, have been successfully used in several audio tasks

such as acoustic scene classification [5, 6] and sound event detection [6], achieving state of the art results. They were also used to tackle problems similar to QbyE KWS such as Spoken Term Detection (STD) [7] and Keyword Spotting [8, 9]. A CNN approach using bottleneck features was proposed in [10] and was shown to outperform Dynamic Time Warping (DTW) approaches in low-resource QbyE STD tasks.

In this work, we introduce an edge friendly CNN architecture inspired by the MLPMixer [11]. The MLPMixer is a recently proposed model that only uses Multilayer Perceptrons (MLPs). Other similar MLP-based models were also proposed [12, 13, 14]. These models achieved competitive performance on the ImageNet [15] dataset. Most recently, MLPMixer was adapted to the QbyE open-vocabulary keyword spotting problem and outperformed some of the state-of-the-art CNN and RNN models [16]. The MLPs can be viewed as a special case of CNNs (e.g. with 1×1 convolutions). However, since CNNs provide more flexibility, we first construct a convolutional equivalent version of MLPMixer called ConvMixer. We then studied the effectiveness of adding different mechanisms that were generally used in light-weight CNNs to the ConvMixer. For example, we explored adding depthwise convolution, Squeeze and Excitation (SE) layer [17] that is successfully used in MobileNet [18], and EfficientNet [19] family of models. Further more, we replaced the convolution with dynamic convolution [20] to our small footprint ConvMixer. By adding dynamic convolution, we are able to boost the performance of our model by a large margin while adding a small number of extra parameters and multiply-accumulates (MACs).

In summary, our contributions are as follows: (1) We propose an efficient and effective ConvMixer model with dynamic convolution. (2) We compare the performance of our proposed model against strong baselines on the publicly available Hey-Snips and Hey-Snapdragon datasets in various far-field and non far-field settings. (3) We showcase the impact of dynamic convolution and its attention mechanism on the performance of our model using ablation experiments. (4) We discussed the importance of having an evaluation system that accurately reflects the model's performance, and we detailed our evaluation methods which can serve as a discussion point for future work.

2. Methods

We use an 81-dimensional Mel-frequency Cepstral Coefficients (MFCCs) with 128 mel filterbanks extracted from 1s long audio with window of 25 ms and stride of 12.5 ms as input to the model. This will give us features with the following dimensions: 1 channel \times 81 MFCCs \times 81 time steps. We apply CMVN on the temporal dimension to normalize the MFCC features. Since the channel dimension is 1, we will omit it in this

*Authors contributed equally

paper for simplicity.

The outline of the proposed system is depicted in Figure 1. We focus on optimizing the encoder network so the keyword’s acoustic information can be mapped to a low dimensional embedding. A simple fully connected linear classifier is added on top of the embeddings and used only during training.

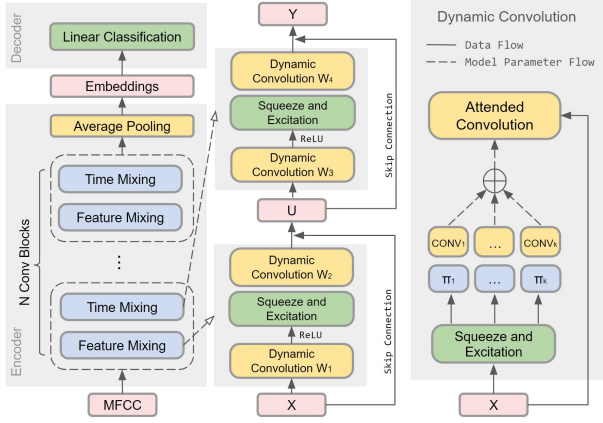


Figure 1: Encoder-decoder system architecture with dynamic ConvMixer blocks

2.1. MLP Mixer vs. ConvMixer

Our convolution architecture is inspired by the recently proposed MLP Mixer architecture [11], which replaces the attention mechanism and positional encoding in ViT [21] with pure MLPs and achieves competitive performance on many vision tasks. In this paper, we implement a convolutional version of the MLP Mixer called ConvMixer. As we know, a linear layer can be seen as 1×1 convolution or single-channel depthwise convolution of the entire receptive field [11]. We opt for the first interpretation and thus, in our implementation, we first consider the feature dimension m from the input $\mathbf{X} \in \mathbb{R}^{m \times n}$ as the channel dimension, and apply two 1×1 convolutions (\mathbf{W}_1 and \mathbf{W}_2) to \mathbf{X} to form the feature mixing output $\mathbf{U} \in \mathbb{R}^{m \times n}$. We then consider the temporal dimension from \mathbf{U} as the input channels to another two 1×1 convolutions (\mathbf{W}_3 and \mathbf{W}_4) to form the time mixing output $\mathbf{Y} \in \mathbb{R}^{m \times n}$. Formally, we used the following equations:

$$\begin{aligned} \mathbf{U} &= \mathbf{X} + \text{SE}(\sigma(\mathbf{X} * \mathbf{W}_1)) * \mathbf{W}_2 \\ \mathbf{Y} &= \mathbf{U} + (\text{SE}(\sigma(\mathbf{U}^T * \mathbf{W}_3)) * \mathbf{W}_4)^T \end{aligned} \quad (1)$$

For *feature mixing*, we first use $\mathbf{W}_1 \in \mathbb{R}^{m \times g}$ to change the input channel from m to g , then use $\mathbf{W}_2 \in \mathbb{R}^{g \times m}$ to project the channel from g back to m , so that the residual connection can be applied. We used ReLU as the activation function σ . We then take the transpose of the output \mathbf{U} from feature mixing, and perform similar operations on it to generate the *time mixing* output \mathbf{Y} . Concretely, $\mathbf{W}_3 \in \mathbb{R}^{n \times h}$ projects the channel dimension from n to h and $\mathbf{W}_4 \in \mathbb{R}^{h \times n}$ projects it back to n . We used the modular design from the MLP Mixer to keep the model size small.

In addition, between two convolution layers in each mixing block, we added an SE layer to improve model performance. SE can be viewed as a channel attention mechanism. It first uses global average pooling to reduce the feature map to a singular value on each channel. This changes the feature map size from

$C \times F$ to $C \times 1$, where C is the channel dimension and F is the feature dimension. It then uses a Fully Connected (FC) layer to project the “squeezed” feature $C \times 1$ to a bottleneck hidden layer with size C/γ , where γ is a reduction factor. We choose $\gamma = 4$ (the default value for the SE layer) in our architecture. Finally, it uses another FC layer to project the hidden space back to $C \times 1$. We use ReLU activation after the first FC and sigmoid activation after the second FC. This output is then used to scale the channels of the input to SE layer. Note that we removed LayerNorm from the original MLP Mixer since our experiments showed that removing it results in better performance for our task (See section 4.1 for details).

2.2. Dynamic Convolution

Dynamic convolution was proposed by [20] in the stream of dynamic neural networks [22, 23]. It improves the performance of small vision models by introducing a small computational overhead. Dynamic convolution applies the attention mechanism to convolution kernels. Specifically, it replaces the traditional single convolution kernel \mathbf{W} with K convolution kernels $\tilde{\mathbf{W}}_k$ and aggregates them together using softmax weights. We used $K = 4$ convolutional kernels in our model. The weights for each kernel is dependent on the input, so different inputs will result in different weights for each kernel. The weights and kernels are aggregated dynamically as follows: $\tilde{\mathbf{W}} = \sum_k \pi_k(\mathbf{X}) * \tilde{\mathbf{W}}_k$. Extra MACs are introduced by computing the attentions $\pi_k(\mathbf{X})$ and aggregating the kernels, and extra parameters are introduced by the K parallel convolution kernels. It is important to note that the attention weights are calculated using the computationally efficient SE layer (The same SE concept mentioned in Section 2.1 but with softmax instead of sigmoid). Therefore, given that our kernel size is 1×1 , this only adds 33.8K parameters and 672K MACs. We plug in the dynamic convolution to our ConvMixer model by replacing all the \mathbf{W}_i with $\tilde{\mathbf{W}}_i$ ($i = 1, 2, 3, 4$) in Equation 1.

3. Experiments

The models were trained using the LibriSpeech [24] dataset following the procedure in [4]. We used the most frequent 10K words from LibriSpeech as our output classes.

We followed the training procedure outlined in [20] and decayed the attention temperature from 34 to 1 in steps of 3 every epoch. During inference, we observed that adding the temperature parameter serves as a regularization mechanism and actually improves the performance of the model. This can be attributed to the fact that our testing task is different from the one we optimize for during training. We observe that dividing all attention scores by a temperature of 34 before computing the softmax in the SE layer helps improve generalization.

3.1. Evaluation Dataset

The models are evaluated on two publicly available datasets: Hey-Snips [9] and Hey-Snapdragon [25]. Hey-Snips dataset contains positive samples of a single keyword: “hey snips”, we used all the 40 speakers with at least 10 utterances from its test set as positive samples for evaluation. There are four keywords in Hey-Snapdragon dataset: “hey android”, “hey snapdragon”, “hi lumina” and “hi galaxy”. The keyword “hi galaxy” has 934 utterances from 42 speakers while the others have 1,112 utterances each from 50 speakers. For each keyword from each speaker, three utterances were randomly selected for enrolling and the rest are used for positive querying. The negative sam-

ples in our evaluation is the negative Hey-Snips test set. It contains about 20K utterances of general sentences totaling 23.30 hours.

For both positive and negative queries, we added “non far-field” and “far-field” settings to evaluate our model. For each dataset, we generate 3 scenarios (clean, 10dB, and 6dB SNR) under “non far-field” and “far-field” conditions. This results in 6 situations in total for positive queries. To simplify the evaluation, negative samples were augmented into two scenarios: “non far-field” and “far-field”. For the “non far-field” negative set, we first added noise to the negative samples at 6dB and 10dB, then we randomly picked half samples from both SNRs and used the mixed set as our negative set. Similarly, the same procedure was followed for the negative set of the “far-field” condition but with far-field effects added on top of the noise.

For the non far-field condition, we added noise from the Microsoft Scalable Noisy Speech Dataset (MS-SNSD) [26] to the keywords with 10dB and 6dB SNR. For “far-field” situation, we followed the kaldi [27] implementation of [28] to add far-field effects using the following equation:

$$x_r[t] = x[t] * h_s[t] + \sum_i n_i[t] * h_i[t] \quad (2)$$

where $x_r[t]$ represents simulated far-field speech, $x[t]$ and $h_s[t]$ represent the audio signal and the corresponding Room-Impulse-Response (RIR), $n_i[t]$ and $h_i[t]$ represent a point-source noise and its corresponding RIR. Specifically, we randomly choose one RIR provided from [28] as h_s to add to the input audio signal x , and another random RIR as h_i to add to a point-source noise sampled from the MS-SNSD noise folder.

The noises and RIRs we added to the evaluation dataset are *noise-test* from MS-SNSD and *real_rir* from [28]. Note that for training, we used *noise-train* from MS-SNSD and *simulated_rir* from [28].

3.2. Evaluation Method

During our evaluation, we observed that the Voice Activity Detection (VAD) filtered data reflected the model’s performance more accurately since some models are able to obtain good results by being biased by the amount of silence in the audio. For example, we found enroll and positive queries tended to have more silence in the beginning and the end of the recorded audio clip than the negative queries due to the methodology with which data was recorded. A model may show good results by only comparing the silence portion between the enrollment and the query which results in subpar performance in a real-world streaming setting. Therefore, to emulate real-world conditions, we used a VAD model [29] to remove the silent portions of audio in the Hey-Snips and Hey-Snapdragon datasets. Using the VAD reduces the average utterance length from 3.96s to 1.69s for positive Hey-Snips queries and from 1.10s to 1.01s for the Hey-Snapdragon queries. This results in all enrollments and positive queries in our evaluation data to be shorter than 2s. Hence, we limit the size of the enrollment audio to 2s and have an audio buffer that is 2s long for storing incoming audio during the query phase. Since the model was trained on 1s long audio, we chunk the input audio into 1s long chunks with a stride of 100 ms. For example, a 2s long audio produces 11 1s chunks which are then fed to the model to produce 11 embeddings q . To match the size between the enrollments and the query, we simply convolve the enrollment embeddings e with the query embeddings q . The cosine distance between the e and q is then computed. Typically, multiple distances will

be generated from each enrollment e_i . These distances can be aggregated (we call this micro-aggregation) to form a final distance d_i for each e_i . On top of that, another aggregation (we call this macro-aggregation) can be applied to d_i to generate the final decision value d . From our experiments, we found that mean macro-aggregation and minimum micro-aggregation gave the best results among the four different combinations of mean and minimum functions for macro and micro-aggregation.

In order to make the negative samples simulate the streaming setting without concatenating it, we extract 2s long windows from the query samples. Any samples shorter than 2s are randomly zero padded on both sides. For the 10dB and 6dB environments, we add noise to the extracted query at the corresponding SNR.

4. Results

4.1. Model Construction

In this section, we experimentally show how different components were assembled together in our DyConvMixer. We use the convolutional equivalent of the MLPMixer without LayerNorm as our base model. We also experimented with components such as depthwise convolution and SE by adding them between the two convolutional layers in the conv blocks. These components were successfully used in MobileNet and EfficientNet and resulted in improved performance. The depthwise convolution layer used the same number of input and output channels output by the first convolutional layer in the conv block. In addition, we investigated dynamic convolution which was specifically designed for improving the performance of small models. Added to that, we also tested the effectiveness of LayerNorm since it was used in the original MLPMixer architecture.

As shown in Table 1 (for simplicity, we report the averaged FRR across clean, 10dB and 6dB here), surprisingly, adding LayerNorm (LN) and depthwise convolution (Depth) to the ConvMixer degraded its performance. On the other hand, adding an SE block improved the performance in all conditions. Among all the components we added, dynamic convolution (Dy) shows significant performance improvements. Based on our observation of the experiments, we added both Dy and SE components together with the ConvMixer to form our Dy-ConvMixer as illustrated in Figure 1. Comparisons in later sections are all based on this architecture.

Table 1: FRR (%) at 0.3 FAs per hour for various components

Model	Non far-field		Far-field	
	Hey-Snips	Snapdragon	Hey-Snips	Snapdragon
Base	23.69	15.06	50.95	41.60
+ LN	24.05 _(+0.36)	24.00 _(+8.94)	48.21 _(-2.74)	47.28 _(+5.68)
+ Depth	33.71 _(+10.02)	21.46 _(+6.40)	55.12 _(+4.17)	46.60 _(+5.00)
+ SE	20.64 _(-3.05)	14.29 _(-0.77)	40.71 _(-10.24)	39.56 _(-2.04)
+ Dy	4.27 _(-19.42)	6.58 _(-8.48)	19.76 _(-31.19)	23.25 _(-18.35)
+ Dy + SE	3.09 _(-20.60)	5.51 _(-9.55)	13.69 _(-37.26)	23.55 _(-18.05)

4.2. Benchmark Comparison

We selected three categories of baseline models to compare with DyConvMixer: (1) An RNN-based GRU-ATTN [4] which also reported open vocabulary keyword spotting performance on Hey-Snips. (2) MobileNet and EfficientNet: MobileNetV2[30] and V3[31] have been adapted to many audio problems and achieved state-of-the-art [32, 6] results. From the EfficientNet family, we included B0 and B1[19] as they have relatively small

model sizes. (3) ViT, which gained popularity in vision tasks, and was also adapted to the keyword spotting task [33]. Table 2 shows that DyConvMixer is significantly smaller compared than the baseline models. In particular, it has less than half the number of parameters in GRU-ATTN and less than half the number of MACs in MobileNetV3.

Table 2: Number of parameters (M) and MACs (M)

Model	Params	MACs	Model	Params	MACs
GRU-ATTN [4]	0.55	41.23	EfficientNetB1 [19]	6.51	58.66
MobileNetV2 [30]	2.22	29.22	ViT [21]	0.96	77.06
MobileNetV3 [31]	2.97	22.24	ConvMixer	0.15	10.30
EfficientNetB0 [19]	4.01	39.06	Ours	0.19	10.97

Tables 3 and 4 show the False Rejection Rate (FRR) at 0.3 False Acceptance per hour under “non far-field” and “far-field” conditions respectively. Our model shows comparable results to MobileNetV3 and EfficientNetB1 under the “non far-field” setting. For the “far-field” setting, DyConvMixer gives much best results on the Hey-Snips dataset, but is outperformed by MobileNetV3 by about 6% on the 10dB and 6dB noisy Hey-Snapdragon dataset.

Table 3: FRR (%) at 0.3 FAs per hour for clean, 10db and 6db “non far-field” condition.

Model	Hey-Snips			Snapdragon		
	clean	10dB	6dB	clean	10dB	6dB
RNN-ATTN [4]	1.43	5.36	11.07	1.32	13.45	26.12
MobileNetV2 [30]	1.07	4.64	12.50	1.61	6.25	12.47
MobileNetV3 [31]	0.36	5.36	16.43	0.93	4.23	10.05
EfficientnetB0 [19]	0.71	5.71	7.14	2.24	7.72	14.66
EfficientnetB1 [19]	0.36	3.93	10.71	1.17	5.11	11.83
ViT [21]	6.07	11.79	17.14	7.43	23.63	35.24
Ours	0.71	2.50	6.07	0.87	4.85	10.80

Table 4: FRR (%) at 0.3 FAs per hour for clean, 10dB and 6dB “far-field” condition.

Model	Hey-Snips			Snapdragon		
	clean	10dB	6dB	clean	10dB	6dB
RNN-ATTN [4]	8.57	22.50	29.29	12.68	38.75	51.79
MobileNetV2 [30]	15.36	27.86	37.86	20.41	33.57	41.88
MobileNetV3 [31]	13.21	27.86	37.50	10.26	20.35	31.45
EfficientnetB0 [19]	11.79	24.64	34.64	14.68	31.05	43.33
EfficientnetB1 [19]	12.86	22.50	34.29	13.87	28.74	40.25
ViT [21]	13.21	25.71	35.24	26.31	52.19	60.21
Ours	6.07	13.57	21.43	7.13	26.25	37.26

4.3. Ablation Study

In this section, we study the impact of the attention mechanism in dynamic convolution as well as the performance improvements resulting from adding dynamic convolution to the ConvMixer.

4.3.1. Attention Mechanism

To understand the value that the softmax attention mechanism brings to dynamic convolution, we experimented with the original softmax along with 2 other aggregations to the convolutional kernels. The first aggregation is to simply average the kernel weights so the aggregation no longer depends on the input. The second aggregation is by selecting the convolutional kernel with the maximum softmax weight. This will inform us whether the

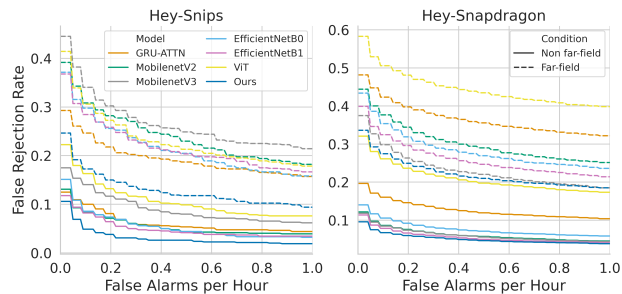


Figure 2: ROC on the Hey-Snips and Hey-Snapdragon datasets under the non far-field and far-field settings

attention mechanism is performing kernel selection. Table 5 shows that softmax aggregation outperforms mean and max aggregation. This indicates that the attention mechanism is able to adapt to its input effectively.

Table 5: FRR (%) at 0.3 FAs per hour for various aggregation mechanisms.

Agg	Non far-field		Far-field	
	Hey-Snips	Snapdragon	Hey-Snips	Snapdragon
Mean	7.50	7.06	17.74	30.08
Max	9.41 _(+1.91)	12.88 _(+5.82)	27.02 _(+9.28)	37.70 _(+7.62)
Softmax	3.09 _(-4.41)	5.51 _(-1.55)	13.69 _(-4.05)	23.55 _(-6.53)

4.3.2. Dropping Dynamic Convolution

In our network, there are two convolution operations per block (excluding the SE layer), we experimented with replacing either convolution kernel with dynamic convolution and observe how it impacts performance. Table 6 shows that replacing a single convolution layer with dynamic convolution results in a large boost and replacing both convolutions performs best.

Table 6: FRR (%) at 0.3 FAs per hour for dynamic conv layers

DyC1	DyC2	Non far-field		Far-field	
		Hey-Snips	Snapdragon	Hey-Snips	Snapdragon
No	No	20.36	14.29	40.71	39.56
Yes	No	3.81 _(-16.55)	7.15 _(-7.14)	15.00 _(-25.71)	38.58 _(-0.98)
No	Yes	6.31 _(-14.05)	8.11 _(-6.18)	15.24 _(-25.47)	26.10 _(-13.46)
Yes	Yes	3.09 _(-17.27)	5.51 _(-8.78)	13.69 _(-27.02)	23.55 _(-16.01)

5. Conclusion and Future Work

This paper presents DyConvMixer, an efficient and effective model on the QbyE KWS task. By constructing a convolutional equivalent of the MLP Mixer architecture and adding SE and dynamic convolution, we obtain a competitive model with less than 200K parameters and around 10M MACs. Compared to other baselines, our model performs the best on most environments and has strong performance on the noisy Hey-Snapdragon dataset while having less than half the number of parameters and MACs. We show that replacing convolutional kernels with dynamic convolution results in a large boost in performance while only adding 27% more parameters and 6.5% more MACs. We detailed our evaluation pipeline and benchmarked our model on publicly available datasets. Finally, we conducted an ablation study to showcase the effectiveness of different components in dynamic convolution.

6. References

- [1] G. Chen, C. Parada, and T. N. Sainath, "Query-by-example keyword spotting using long short-term memory networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5236–5240.
- [2] N. Sacchi, A. Nanchen, M. Jaggi, and M. Cernak, "Open-vocabulary keyword spotting with audio and text embeddings," in *INTERSPEECH 2019-IEEE International Conference on Acoustics, Speech, and Signal Processing*, no. CONF, 2019.
- [3] T. Bluche, M. Primet, and T. Gisselbrecht, "Small-footprint open-vocabulary keyword spotting with quantized lstm networks," *arXiv preprint arXiv:2002.10851*, 2020.
- [4] J. Huang, W. Gharbieh, H. S. Shim, and E. Kim, "Query-by-example keyword spotting system using multi-head attention and soft-triple loss," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6858–6862.
- [5] B. Kim, S. Yang, J. Kim, and S. Chang, "QTI submission to DCASE 2021: Residual normalization for device-imbalanced acoustic scene classification with efficient design," DCASE2021 Challenge, Tech. Rep., June 2021.
- [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," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.
- [7] D. Ram, L. M. Werlen, and H. Bourlard, "Cnn based query by example spoken term detection," in *Interspeech*, 2018, pp. 92–96.
- [8] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5484–5488.
- [9] A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Poumeyrol, and T. Lavril, "Efficient keyword spotting using dilated convolutions and gating," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6351–6355.
- [10] D. Ram, L. Miculicich, and H. Bourlard, "Neural network based end-to-end query by example spoken term detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1416–1427, 2020.
- [11] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [12] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, G. Izacard, A. Joulin, G. Synnaeve, J. Verbeek *et al.*, "Resmlp: Feedforward networks for image classification with data-efficient training," *arXiv preprint arXiv:2105.03404*, 2021.
- [13] H. Liu, Z. Dai, D. So, and Q. Le, "Pay attention to mlps," *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [14] X. Ding, C. Xia, X. Zhang, X. Chu, J. Han, and G. Ding, "Repmlp: Re-parameterizing convolutions into fully-connected layers for image recognition," *arXiv preprint arXiv:2105.01883*, 2021.
- [15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [16] J. Huang, W. Gharbieh, Q. Wan, H. S. Shim, and C. Lee, "Qbye-mlpmixer: Query-by-example open-vocabulary keyword spotting using mlpmixer," *arXiv preprint arXiv:2206.13231*, 2022.
- [17] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.
- [18] 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 preprint arXiv:1704.04861*, 2017.
- [19] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.
- [20] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, "Dynamic convolution: Attention over convolution kernels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 030–11 039.
- [21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
- [22] L. Liu and J. Deng, "Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
- [23] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 409–424.
- [24] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.
- [25] B. Kim, M. Lee, J. Lee, Y. Kim, and K. Hwang, "Query-by-example on-device keyword spotting," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 532–538.
- [26] C. K. Reddy, E. Beyrami, J. Pool, R. Cutler, S. Srinivasan, and J. Gehrke, "A scalable noisy speech dataset and online subjective test framework," *Proc. Interspeech 2019*, pp. 1816–1820, 2019.
- [27] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldı speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.
- [28] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5220–5224.
- [29] S. Team, "Silero vad: pre-trained enterprise-grade voice activity detector (vad), number detector and language classifier," <https://github.com/snakers4/silero-vad>, 2021.
- [30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.
- [31] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.
- [32] R. Tang, J. Lee, A. Razi, J. Cambre, I. Bicking, J. Kaye, and J. Lin, "Howl: a deployed, open-source wake word detection system," *arXiv preprint arXiv:2008.09606*, 2020.
- [33] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword Transformer: A Self-Attention Model for Keyword Spotting," in *Proc. Interspeech 2021*, 2021, pp. 4249–4253.