Teaching keyword spotters to spot new keywords with limited examples

Abhijeet Awasthi¹,², Kevin Kilgour¹, Hassan Rom¹

¹Google Research, Switzerland
²Indian Institute of Technology Bombay, India

awasti@cse.iitb.ac.in, {kkilgour,hassanrom}@google.com

Abstract

Learning to recognize new keywords with just a few examples is essential for personalizing keyword spotting (KWS) models to a user’s choice of keywords. However, modern KWS models are typically trained on large datasets and restricted to a small vocabulary of keywords, limiting their transferability to a broad range of unseen keywords. Towards easily customizable KWS models, we present KeySEM (Keyword Speech EMbedding), a speech embedding model pre-trained on the task of recognizing a large number of keywords. Speech representations offered by KeySEM are highly effective for learning new keywords from a limited number of examples. Comparisons with a diverse range of related work across several datasets show that our method achieves consistently superior performance with fewer training examples. Although KeySEM was pre-trained only on English utterances, the performance gains also extend to datasets from four other languages indicating that KeySEM learns useful representations well aligned with the task of keyword spotting. Finally, we demonstrate KeySEM’s ability to learn new keywords sequentially without requiring to re-train on previously learned keywords. Our experimental observations suggest that KeySEM is well suited to on-device environments where post-deployment learning and ease of customization are often desirable.

Index Terms: Spoken Term Detection, Keyword Spotting, Wakeword Recognition, Hotword Recognition, Speech Embeddings

1. Introduction

Keyword spotting (KWS) also known as spoken term detection or Keyphrase / Wakeword / Hotword recognition is a task of identifying whether a speech segment contains an utterance of a target keyword. It is typically formulated as a task of classifying short speech segments into one of the pre-defined keywords. Recent advances in neural architectures for on-device, low-footprint KWS models [1][2][3][4] have made their deployment possible in various commercial products such as Google Home, Apple HomePod, and Amazon Echo. KWS models are typically trained on thousands of hours of in-house datasets [3], often restricted to examples from a very particular set of keywords, limiting their transferability to a larger domain of unseen keywords. However, there has been a rising focus towards on-device personalization of speech recognition technology [5][6][7]. Enabling keyword spotters to learn from just a few examples would allow easy registration of user-specified keywords on personal devices.

Towards this goal, we present KeySEM, a speech embedding model which allows for more accurate KWS models to be learned from fewer training examples. KeySEM maps short spans of speech utterances to fixed dimensional vectors (embeddings). KeySEM is pre-trained on a classification task of recognizing 15K different keywords extracted from the LibriSpeech corpus [8]. We show that pre-training KeySEM to simultaneously discriminate across a wide range of 15K keywords for classification allows it to learn more discriminative representations of speech segments. In comparison to recent self-supervised pre-training methods for learning general purpose speech representations [9][10], our supervised pre-training method is directly aligned with the task of keyword spotting. We conduct extensive comparisons between our method and the multi-task supervised pre-training method of Lin et al. [2], where each classification head learns to discriminate across just 40 keywords. Despite using three orders of magnitude less data than Lin et al. [2] during pre-training, we achieve superior performance on several downstream KWS datasets, suggesting that our pre-training method offers better representations by exploiting a much larger vocabulary of keywords.

Section 3 describes the architecture of KeySEM and the pre-training details. Section 4 describes our experiments on datasets spanning English and four other languages (Japanese, Portuguese, Polish and Esperanto). Compared to recent KWS models, utilizing KeySEM offers up to 61% absolute improvements in accuracy, being particularly effective when training data is limited. Interestingly, even though we pre-train KeySEM only on English speech, performance gains are also observed on datasets from the four other languages, suggesting that representations learned while pre-training KeySEM generalize well for the task of keyword spotting.

2. Related Work

KWS Models: Keyword spotting is generally formulated as a task of classifying fixed length speech segments into a known vocabulary of keywords. Modern KWS models [1][4][11][12][13][14][15][16] are often based on neural network classifiers with audio features like Mel-frequency Cepstral Coefficients, extracted from the speech signal as input. Sainath and Parada [1] were among the first to explore CNNs for KWS under memory and compute constrained settings. More recent work by Rybakov et al. [17] provides us with an extensive comparison of a diverse range of KWS architectures.

KWS with limited data: Neural network based models require thousands of training examples per keyword to learn useful KWS models. To address this problem, researchers have tried several approaches, including initializing a part of KWS models with weights of an ASR model [18], data-augmentation [3], meta-learning [19], few-shot learning [20] and multitask learning [2]. Gao et al. [3] propose data augmentation techniques such as the addition of reverberation and noise to simulate far-field speech. SpecAugment [21] is another common data-augmentation technique used in various KWS models [11][17]. Chen et al. [19] extend the few-shot meta-learning framework MAML [22] to KWS. Parnami and Lee [20] explore proto-
typical networks \[23\] to learn speech embeddings for improved few-shot transfer. Bluche and Gisselbrecht \[13\] propose a few-shot learning method for adaptation to new keywords. However, they assume access to a grapheme-to-phoneme lexicon, making their approach language-dependent and challenging to extend to low-resource languages where the required lexicons may not be readily available. In contrast, our method does not assume access to such resources and works well even for the languages unseen during the pre-training phase.

The most similar method to our work is that of Lin et al. \[2\], where a speech embedding model is learned from 111K hours of speech extracted from YouTube. Their embedding model is pre-trained in a multitask learning setup where it is shared across 125 keyword spotters, each with a different keyword vocabulary of size 40. In contrast, our embedding model is pre-trained on a single task of classifying speech segments into a much larger vocabulary of 15K keywords, thereby learning more discriminative features useful for recognizing keyword utterances. Our embedding model offers significantly better downstream performance across several datasets despite using three orders of magnitude less data during pre-training.

3. Our Method

\subsection*{3.1. Architecture details}

The embedding model \(F_\theta\) uses a CNN based architecture similar to \[2\]. It assumes a 2s long speech segment as input which is converted to 40 dimensional log-mel features obtained using window size of 25 ms and a step size of 10 ms covering the frequency range from 60 Hz to 7800 Hz. The log-mel features are then passed to the CNN architecture which consists of 6 convolutional blocks. The first five convolutional blocks each consist of 5 layers: 4 alternating a 1x3 and 3x1 convolutions, followed by a maxpool layer. After the fifth block the frequency dimension becomes 1. The final block comprises two consecutive layers each composed of a 5x1 convolution and an average-pool layer. Starting with 24 channels in the first block, we increase the number of channels by 24 in each new block until a maximum of 96 is reached. Ultimately, the embedding model maps 2s long speech segments to 96-dimensional feature vectors. We refer the reader to Figure 1 in \[2\] for a view of the embedding model’s architecture.

\subsection*{3.2. Dataset extraction}

Pre-training KeySEM requires keyword utterances from a large representative vocabulary \(V\). The publicly available LibriSpeech corpus \[8\], together with community created forced-alignments \[24, 26\], provide us with many easily extractable keywords. As keywords, we use \(n\)-grams (\(n <= 5\)) that are at least 10 characters long and have at least 10 occurrences in the train split of LibriSpeech corpus, giving us over 15.2K different keywords. The extracted keyword utterances constitute 250hrs of speech. Keyword utterances in the dev split of LibriSpeech serve as the dev set during pre-training.

Prior datasets used for benchmarking KWS models’ performance involve only a small number of keywords (often less than 20). To evaluate KWS models on a broader range of unseen keywords, we hold out 150 keywords with at least 100 occurrences and their associated utterances from the pre-training dataset. In Section 4.1 we report the performance of various models on this held out dataset, which we refer to as LibriKWS.

4. Experiments

Our experiments on three datasets spanning five languages show that KeySEM helps tremendously when training data is limited, even for the languages not seen by KeySEM during pre-training. This section provides a brief description of the datasets followed by the experimental results. We plan to release the datasets and pre-trained models.

\subsection*{4.1. Datasets}

\textbf{Google Speech Commands (V2)} \[27\] is commonly used for benchmarking KWS models. It consists of utterances for 35 different words like \{Up, Down, Left, Right, \ldots\}. 25 of these words are grouped into a single category representing the negative class. We use the default train, dev, and test splits as available on the tensorflow website \[28\], consisting of 85.5K, 10.1K and 4.9K utterances respectively.
Figure 1: PCA projections of embeddings obtained from (a) KeySEM, (b) Lin et al. [2], and (c) TRILL for utterances of “good morning”, “tried to speak”, “trying to keep”, and “right angles”. Note: Embedding models were NOT trained on these keywords.

LibriKWS: As described in Section 3.2, LibriKWS contains utterances for the 150 keywords not used to pre-train KeySEM. We use 90 examples per keyword for the train split and 10 examples per keyword for the dev split. For evaluation, we construct a ‘test-clean’ set by pooling utterances extracted from the ‘test-clean’ and ‘dev-clean’ splits of LibriSpeech. Similarly, a ‘test-other’ set is constructed by pooling utterances extracted from the ‘test-other’ and ‘dev-other’ splits. The ‘test-other’ split is expected to be more challenging than ‘test-clean’. The design of the original LibriSpeech dataset [8] ensures zero speaker overlap across between the train and test splits. To the best of our knowledge, no open-source KWS dataset provides more target classes than LibriKWS.

Common Voice [29]: To evaluate the effectiveness of KeySEM on non-English keywords, we use the single word spoken digit recognition dataset for four languages, including Japanese (ja), Esperanto (eo), Polish (pl), and Portuguese (pt). The number of target keywords varies between 12 and 14. We use the default train, test, and dev splits consisting an average of 335, 311, and 306 utterances respectively.

To facilitate comparisons with prior work and given no severe class imbalances in the test sets, we report accuracy as a performance metric.

4.2. Results

Overall Comparisons: Table 1 presents our method’s overall comparisons with recent KWS models after training each model on the entire training set of the respective datasets. Our comparisons include MatchBoxNet [11] composed of convolutional layers with residual connections, with roughly 85K weights. MHAtt-RNN represents the most accurate model from [17] with roughly 765K weights, composed of multi-head attention layers [30] followed by an RNN layer. For MatchBoxNet and MHAtt-RNN, we use the implementation provided by [17] [31]. MTLEmb refers to KWS models using convolutional heads on top of the speech embedding model described in [2], with roughly 400K weights in the embedding model and 50K weights in the convolutional head. KeySEM has roughly 410K weights and requires just 96 additional weights per keyword in the linear classification layer. We experiment with fixing (fix) and finetuning (FT) the weights of the embedding models during training. KeySEM (rand) refers to the KWS model obtained by training a randomly initialized KeySEM architecture from scratch. We observe that utilizing a pre-trained KeySEM provides the best results across all the datasets. Keeping KeySEM’s weights fixed while training KWS models also yields competitive results and requires only 96 trainable parameters per keyword. Since the train-split of LibriKWS contains only 90 examples per keyword, fixing KeySEM’s weights also helps avoid over-fitting and yields superior results than finetuning the entire model.

Limited data experiments: Table 2 reports the performance of various KWS models when trained on just 5 randomly selected examples per keyword. For the Speech Commands dataset, we retain 15 training examples for the negative class. Models like Matchbox and MHAtt-RNN utilize SpecAugment [21] for data augmentation to overcome data scarcity and overfitting. From Table 2 we observe that utilizing pre-trained embedding models like Lin et al. [2] or KeySEM with fixed weights provides consistent gains over fine-tuning or training the models from scratch using data augmentation techniques. KeySEM offers absolute gains between 7% to 61% over Lin et al. [2]’s method (MTLEmb). Surprisingly, the gains offered by KeySEM also extend well to datasets from four other languages, including Japanese (ja), Esperanto (eo), Polish (pl), and Portuguese (pt), even though KeySEM is pre-trained only on English utterances. Figure 2 provides a closer comparison between KeySEM and MTLEmb. Along the x-axis, we increase the number of training examples per keyword and report the test accuracy on the y-axis for the Speech Commands dataset. With less than 10 training examples per keyword, training with KeySEM’s weights held fixed offers superior performance over other methods. The performance difference across various methods diminishes with increasing amounts of training data. For 1000 examples per keyword, fine-tuning the entire KeySEM offers the highest accuracy of 97.5%.

| Method     | SC LK-c | LK-o | ja | eo | pl | pt |
|------------|---------|------|----|----|----|----|
| Matchbox   | 98.0    | 97.3 | 89.8 | 76.0 | 88.5 | 85.0 | 84.3 |
| MHAtt-RNN  | 98.0    | 99.7 | 95.3 | 86.0 | 87.0 | 87.3 | 79.3 |
| MTLEmb (fix) | 96.6 | 95.1 | 87.2 | 86.7 | 87.4 | 89.5 | 82.6 |
| MTLEmb (FT) | 97.7   | 94.9 | 88.1 | 75.0 | 79.6 | 74.2 | 72.1 |
| KeySEM (rand) | 97.2   | 93.6 | 78.1 | 93.2 | 84.1 | 85.4 | 78.3 |
| KeySEM (fix) | 93.9  | 99.8 | 97.8 | 92.3 | 91.2 | 90.3 | 82.4 |
| KeySEM (FT) | 98.2   | 97.8 | 93.2 | 92.9 | 89.1 | 90.3 | 84.7 |
We use the Speech Commands dataset to explore the ability of getting of previously learned classes \[32, 33\]. In this section, classes sequentially has been shown to cause catastrophic for- 

We have observed that simply training linear classifiers with varying number of training examples, which indicates that their embedding model is pre-trained to work best with convolutional heads. Training a linear layer that requires only 96 trainable parameters per key-

We propose KeySEM, a speech embedding model pre-trained to avoid catastrophic forgetting. We defer exploration of methods like \[2\] [33] to avoid catastrophic forgetting. We defer exploration of methods like \[2\] [33] in the context of KWS to future work.

5. Conclusion

We propose KeySEM, a speech embedding model pre-trained to recognize utterances of a large number of keywords. Learning to discriminate across a wide range of keywords during pre-training allows KeySEM to offer features helpful for the task of Keyword Spotting. On datasets spanning five different languages, KeySEM offers consistent performance gains over several recent methods. When training data is limited, KeySEM achieves superior performance over other methods by training only a linear classifier’s parameters. Finally, considering post-deployment learning in on-device environments, we demonstrate KeySEM’s effectiveness in learning from a stream of data where examples for new classes arrive sequentially.

Table 2: Comparing accuracies of various KWS models on the Speech Commands test set (SC), test-clean (LK-c) and test-other (LK-o) splits of LibriKWS, and the test splits of four selected languages in the Common Voice dataset. **Note:** Models were trained using only 5 randomly chosen examples per keyword from the respective train sets.

| Method         | SC   | LK-c | LK-o | ja  | ee  | pt  | PL  |
|----------------|------|------|------|-----|-----|-----|-----|
| Matchbox       | 45.3 | 3.0  | 2.8  | 48.9| 60.5| 58.3| 36.3|
| MHA-RNN (fixx) | 48.2 | 27.3 | 12.8 | 35.0| 62.5| 68.0| 53.0|
| MTLEmb (FT)    | 79.1 | 43.9 | 33.1 | 76.5| 74.5| 76.3| 71.3|
| MTLEmb (fix)   | 46.8 | 30.1 | 21.4 | 41.3| 48.0| 43.8| 37.6|
| KeySEM (fix)   | 86.5 | 98.5 | 94.2 | 96.2| 97.4| 96.3| 89.3|
| KeySEM (FT)    | 75.5 | 62.7 | 54.4 | 82.1| 92.0| 83.9| 75.3|

Figure 2: Performance comparison between KeySEM and MTLEmb based KWS models on Speech Commands dataset with varying number of training examples per keyword.

Although Lin et al. \[2\] utilize three orders of magnitude more data during pre-training, the two main reasons behind super-

Performance comparison between KeySEM and MTLEmb based KWS models on Speech Commands dataset.

Figure 3: Learning new keywords sequentially.

KeySEM to learn new keywords sequentially. We begin with learning a sigmoid classifier to distinguish between the utter-

Figure 3: Learning new keywords sequentially.
Acknowledgement The authors would like to thank the anonymous reviewers, Andre Perunicic, Dominik Roblek, Matt Sharifi, Marco Tagliasacchi, Karolis Misiusas, Mauro Verzetti, and Vihari Piratla, for their constructive feedback on this work.

6. References

[1] T. N. Sainath and C. Parada, “Convolutional neural networks for small-footprint keyword spotting,” in Sixteenth Annual Conference of the International Speech Communication Association, 2015.

[2] J. Lin, K. Kilgour, D. Roblek, and M. Sharifi, “Training keyword spotters with limited and synthesized speech data,” in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020, pp. 7474–7478.

[3] Y. Gao, Y. Mishchenko, A. Shah, S. Matsoukas, and S. Vitaladevuni, “Towards data-efficient modeling for wake word spotting,” in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020, pp. 7479–7483.

[4] G. Chen, C. Parada, and G. Heigold, “Small-footprint keyword spotting using deep neural networks,” in 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2014, pp. 4087–4091.

[5] K. C. Sim, P. Zadrazil, and F. Beaufays, “An investigation into on-device personalization of end-to-end automatic speech recognition models,” Proc. Interspeech 2019, pp. 774–778, 2019.

[6] “Google AI blog: Improving speech representations and personalized models using self-supervision,” https://ai.googleblog.com/2020/06/improving-speech-representations-and.html (Accessed on 02/12/2021).

[7] I. McGraw, R. Prabhavalkar, R. Alvarez, M. G. Arenas, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Grusenstein, F. Beaufays, and C. Parada, “Personalized speech recognition on mobile devices,” in 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 5955–5959.

[8] 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.

[9] A. Saeed, D. Grangier, and N. Zeghidour, “Contrastive learning of general-purpose audio representations,” in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2021, pp. 3875–3879.

[10] J. Shor, A. Jansen, R. Maor, O. Lang, O. Tuval, F. de Chaumont Quirty, M. Tagliasacchi, I. Shavitt, D. Emanuel, and Y. Haviv, “Towards learning a universal non-semantic representation of speech,” Proc. Interspeech 2020, pp. 140–144, 2020.

[11] S. Majumdar and B. Ginsburg, “Matchboxnet: 1d time-channel separable convolutional neural network architecture for speech commands recognition,” Proc. Interspeech 2020, pp. 3356–3360, 2020.

[12] T. Bluche, M. Primet, and T. Gisselbrecht, “Small-footprint open-vocabulary keyword spotting with quantized lstm networks,” arXiv preprint arXiv:2002.10851, 2020.

[13] L. Lugosch, S. Myer, and V. S. Tomar, “Donut: Ctc-based query-by-example keyword spotting,” arXiv preprint arXiv:1811.10736, 2018.

[14] A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Pounyreyol, and T. Lavri, “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.

[15] 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.

[16] R. Tang, J. Lee, A. Razi, J. Cambre, I. Bicking, J. Kaye, and J. Lin, “Howl: A deployed, open-source wake word detection system,” in Proceedings of Second Workshop on NLP Open Source Software (NLP-OSS), 2020, pp. 61–65.

[17] O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. Laurento, “Streaming keyword spotting on mobile devices,” Proc. Interspeech 2020, pp. 2277–2281, 2020.

[18] T. Bluche and T. Gisselbrecht, “Predicting detection filters for small footprint open-vocabulary keyword spotting,” Proc. Interspeech 2020, pp. 2552–2556, 2020.

[19] Y. Chen, T. Ko, L. Shang, X. Chen, X. Jiang, and Q. Li, “An investigation of few-shot learning in bi-gram term classification,” Proc. Interspeech 2020, pp. 2582–2586, 2020.

[20] A. Parnami and M. Lee, “Few-shot keyword spotting with prototypical networks,” arXiv preprint arXiv:2007.14463, 2020.

[21] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “ScapeAugment: A simple data augmentation method for automatic speech recognition,” Proc. Interspeech 2019, pp. 2613–2617, 2019.

[22] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in International Conference on Machine Learning, PMLR, 2017, pp. 1126–1135.

[23] J. Snell, K. Swersky, and R. Zemel, “Prototypical networks for few-shot learning,” in Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 4080–4090.

[24] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[25] “Word alignments generated by the montreal forced aligner for the librilspeech dataset,” https://github.com/Corenfin/librispeech-alignments (Accessed on 02/19/2021).

[26] M. McAuillife, M. Socolof, S. Mihuc, M. Wagner, and M. Sondergerget, “Montreal forced aligner: Trainable text-speech alignment using kaldi,” in Interspeech, vol. 2017, 2017, pp. 498–502.

[27] P. Warden, “Speech commands: A dataset for limited-vocabulary speech recognition,” arXiv preprint arXiv:1804.03209, 2018.

[28] “Google speech commands dataset,” https://www.tensorflow.org/datasets/catalog/speech_commands (Accessed on 02/18/2021).

[29] R. Ardila, M. Branson, K. Davis, M. Kohler, J. Meyer, M. Henretty, R. Morais, L. Saunders, F. Tyers, and G. Weber, “Common voice: A massively-multilingual speech corpus,” in Proceedings of the 12th Language Resources and Evaluation Conference, 2020, pp. 4218–4222.

[30] 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 31st International Conference on Neural Information Processing Systems, 2017, pp. 6000–6010.

[31] “Tensorflow implementation of matchboxnet,” https://github.com/google-research/google-research/blob/master/kws_streaming/models/ds_kc/nesyet.py (Accessed on 02/18/2021).

[32] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “ICARL: Incremental classifier and representation learning,” in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2017, pp. 2001–2010.

[33] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska et al., “Overcoming catastrophic forgetting in neural networks,” Proceedings of the national academy of sciences, vol. 114, no. 13, pp. 3521–3526, 2017.