Generalized Keyword Spotting using ASR embeddings

Kirandevraj R[1] *Vinod K Kurmi*[1] *Vinay P Namboodiri*[2] *CV Jawahar*[1]

[1] IIIT Hyderabad, India.
[2] University of Bath, UK.
{kirandevraj.r@research., vinodkumarkurmi@research.}iiit.ac.in, vpn22@bath.ac.uk, jawahar@iiit.ac.in

Abstract

Keyword Spotting (KWS) detects a set of pre-defined spoken keywords. Building a KWS system for an arbitrary set requires massive training datasets. We propose to use the text transcripts from an Automatic Speech Recognition (ASR) system alongside triplets for KWS training. The intermediate representation from the ASR system trained on a speech corpus is used as acoustic word embeddings for keywords. Triplet loss is added to the Connectionist Temporal Classification (CTC) loss in the ASR while training. This method achieves an Average Precision (AP) of 0.843 over 344 words unseen by the model trained on the TIMIT dataset. In contrast, the Multi-View recurrent method that learns jointly on the text and acoustic embeddings achieves only 0.218 for out-of-vocabulary words. This method is also applied to low-resource languages such as Tamil by converting Tamil characters to English using transliteration. This is a very challenging novel task for which we provide a dataset of transcripts for the keywords. Despite our model not generalizing well, we achieve a benchmark AP of 0.321 on over 38 words unseen by the model on the MSWC Tamil keyword set. The model also produces an accuracy of 96.2% for classification tasks on the Google Speech Commands dataset.

Index Terms: speech recognition, keyword spotting, low-resource languages

1. Introduction

Keyword Spotting is the problem of determining whether a target term has been uttered in a speech segment. When a term is given in the acoustic form, we call it query-by-example (QbE) KWS. When the keyword set is arbitrary, the network may or may not have seen the keyword during training. This problem of QbE KWS for any arbitrary set is achieved by building an efficient acoustic word embedding for keyword speech segments. Similarity matching is done on these word embeddings of different keywords to find the match. On the other hand, ASR is an acoustic-to-text system where input audio is converted into text transcripts. ASR is continually evolving, with new models advancing state-of-the-art performance. DeepSpeech2 [1] is a speech recognition model that is trained end-to-end with CTC loss on LibriSpeech [2] corpus and has achieved a 6.71 Word Error Rate (WER).

In this paper, we identify that the embeddings from the intermediate layer of DeepSpeech2 ASR can be directly used as an acoustic word embedding for audio keyword segments using the triplet loss and CTC loss. The method is generalized to zero-shot out of vocabulary keywords. In zero-shot setting, the model is asked to predict the keyword classes it has not seen during the training.

Most of the previous works focused on generalization require few samples from all the keywords the model has to predict [3]. They use the target vocabulary samples to train a prediction filter and classify the samples based on the filter predictions. In contrast to those, we directly work on the embedding space so we do not need any training sample for predicting the keywords the model has not seen. We work on zero shot out of vocabulary generalization. Other works [4] require each pair of query and keyword pass through their model together to make predictions. These methods take both query and keyword in the neural network and predict if they are the same or different. But, we get the acoustic word embedding representation for input audio from the model and compare it with the keyword audio embedding that we want to classify. When the similarity between the pair is above a certain threshold, we say that there is a match. We do not need query and keyword pass through the model for each predictions. Our contributions are as follows:

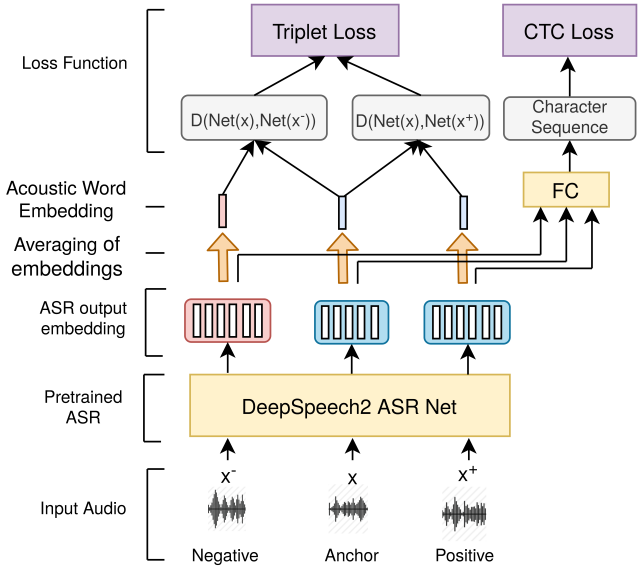

Figure 1: *Generalized Keyword Spotting Embedding Representation: A pretrained DeepSpeech2 ASR Net is trained with CTC loss and triplet loss for obtaining the acoustic word embeddings used for Keyword Spotting*

1. We demonstrate that intermediate representations from ASR when fine-tuned using triplet and CTC loss functions on keyword audio segments extracted from the TIMIT [5] dataset can provide generalized keyword spotting abilities.
2. We then further generalize our approach to low-resource language Tamil from the MSWC [6] dataset. To apply CTC loss we convert the Tamil characters to English using transliteration.
3. Finally, we train our model on Google Speech Com-

mands dataset [7] and compare our results on in-vocabulary classification tasks by using kNN on our acoustic word embeddings and show that they provide comparable in-vocabulary results to the state of the art while having improved generalization abilities.

2. Related Work

Most QbE KWS approaches use dynamic programming to solve the search optimization problem over an input signal or dynamic time warping to allow for duration fluctuations of the target term [8, 9]. Various forms of deep neural networks are used for QbE KWS. Some approaches construct an embedding space, where the query and acoustic examples are projected to this space and is compared [10, 11]. We are approaching this problem in the similar direction.

In recent years, various forms of neural network have proven to be useful for keyword spotting. A convolutional neural network (CNN) is trained on frame level similarities between the posteriors of a spoken query and a test utterance in [12]. A Siamese CNN network is used to generate the embedding space in [13]. A multitask objective of learning acoustic word embedding with triplet loss and cross entropy loss has been explored in [14]. Seq2seq models are used for KWS. The text and audio representation are brought together by the encoder, decoder and a feed forward neural network [15]. ASR posteriors are used to search query in [16]. An encoder network with attention mechanism is used to learn the representation for the query in [17]. An LSTM is trained to discriminate phones with the CTC criterion, and a substring matching algorithm is used to detect the keyword in [18]. The representations are obtained for fixed vocabulary KWS using triplet loss and is classified using k-Nearest Neighbour (kNN) in [19]. Recurrent Neural Networks combined with convolution is used in KWS [20, 21, 22]. Few shot keyword spotting is experimented in [3]. Siamese Recurrent Autoencoders[23] have approached the similar task.

While these contributions have been focused on the task of KWS and have made significant contributions to the task, our contribution differs in terms of being focused on generalizing the KWS ability to a set of arbitrary keywords rather than a fixed small command set. The KWS task used here is the same as that in the literature [24], which is similar to an isolated word recognition task, and it is different from detecting keywords from a continuous speech signal [8, 22].

3. PROPOSED KWS MODEL

3.1. Pretrained ASR with CTC loss

The end-to-end model we use in this work is DeepSpeech2, an acoustics-to-characters system based on a deep neural network. This model is pretrained by OpenSeq2Seq framework [25]. The input to the model is a sequence of audio spectrograms(frequency magnitudes), obtained with a 20ms Hanning window and a stride of 10ms. With a sampling rate of 16kHz, we have 160 dimensional input features. The DeepSpeech2 model has two convolutional layers and five Gated Recurrent Units (GRU). Each convolutional or recurrent layer is followed by batch normalization and a ReLU non-linearity. The model is trained with CTC loss function. The CTC loss function is given as:

$$\mathcal{L}_1 = -\log p(l|x) \tag{1}$$

where the probability of a label sequence l given an input sequence x is defined as:

$$p(\mathbf{l} \mid \mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi \mid \mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} \prod_{t=1}^{T} \mathrm{ASR}_t^K(\mathbf{x})\left[\pi_t\right] \tag{2}$$

where $\mathcal{B}$ removes blanks and repeated symbols, $\mathcal{B}^{-1}$ is its inverse image, $\mathcal{T}$ is the length of the label sequence l, and $\mathrm{ASR}_t^K(\mathbf{x})\left[\pi_t\right]$ is unit j of the model output after the top softmax layer at time t, interpreted as the probability of observing label j at time t.

The acoustic word embedding for the keyword audio segments are obtained from the output of the final GRU layer. The output dimension of the final GRU layer is 1600 for each time instance. We take the average of this representation across time dimension to obtain a fixed representation of size 1600 for variable length audio.

3.2. Adding Triplet loss with CTC loss

As described in [26], a triplet network comprises three instances of the same feed-forward network with shared parameters. We denote the triplet of inputs as $\mathbf{x}$, $\mathbf{x}^+$, $\mathbf{x}^-$ for anchor, positive and negative sample.

The averaged GRU output from $\mathrm{Net}(\mathbf{x})$ is used to calculate the triplet loss. We use cosine similarity as the distance metric. The loss function used to train the triplet network is given as:

$$\mathcal{L}_2\left(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-\right) = \max(\delta[\ \mathrm{Net}(\mathbf{x}), \mathrm{Net}\left(\mathbf{x}^+\right)] - \delta\left[\mathrm{Net}(\mathbf{x}), \mathrm{Net}\left(\mathbf{x}^-\right)\right] + \alpha, 0)$$

where the distance from anchor to the positive sample is minimized, while the distance from anchor to the negative sample is maximized. α denotes the margin between positive and negative sample and δ denotes the distance measurement.

We then append the triplet loss with the calculated CTC loss for backpropagation (Figure 1). Total loss of the model can be give as follows: (λ_1 and λ_2 are the weights given to the CTC and triplet loss)

$$\mathcal{L} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_2$$

3.3. Zero Shot Out-Of-Vocabulary Generalization

This method is easily generalized to Zero-Shot Out Of Vocabulary keywords. This is possible because we operate at the embedding space directly. We obtain the acoustic word embedding for any audio segment from our model as shown in Figure 1. The audio sample can be from keywords the model has or has not seen. During prediction, we convert all the sample audio keyword files into this embeddings space. Then, for every test audio sample, we first obtain the word embedding by passing it through the model. Then we compare the test audio embedding with all the keyword embeddings using cosine similarity. When the similarity is above a certain fixed threshold, we say the keyword and the test audio is a match.

While the cosine similarity comparison method has been the key, the other generalization methods use a modified version with one or few train samples for the keywords. In the work [27], the model requires configuration with a sample keyword audio and respective threshold values for prediction. While our method does not require configuration for each keyword and the threshold is constant. In the work [3], a keyword-specific classification layer is trained with five training samples and uses this classifier for predicting those words. In contrast,

our method does not require any sample for out of vocabulary zero-shot prediction. Other works have focused on fixed vocabulary keyword spotting and require retraining to add new classes.

4. Experiments

4.1. Datasets

TIMIT Dataset: We use TIMIT [5] dataset for training and testing our architecture and the baseline. TIMIT dataset contains approximately five hours of speech. We segment words based on word boundaries instead of voice activity detection in this work. After that, the speech segments are grouped into separate words, and functional words like articles and conjunctions are removed. Finally, words with more than four letters are chosen. The TIMIT dataset has its default train and test split. The train and test splits contain 12261 and 3988 audio segments, consisting of 1645 and 571 unique words. Out of the 571 unique words, 227 words are in vocabulary, and the remaining 344 words are zero-shot out of vocabulary. All the audio files are segmented such that the length of the audio is 1 second.

Google speech command dataset: The data used in this experiment were drawn from the Speech Commands dataset [7]. The dataset consists of 105,829 audio recordings of 35 different English words spoken by 2618 English speakers, representing a general sample of speakers with different accents and speaking styles. Non-keyword or silent samples were not included in the selected data. The training, validation, and test set contains 84843, 9981, and 11005 samples.

Multilingual Spoken Words Corpus (Tamil Split): The MSWC corpus [6] has spoken word segments for keywords in many languages. We selected the Tamil language to understand how the method generalizes to low resource languages as it had very low audio samples for training. The split had a total of 190 keywords and has a total of 1884 audio samples. We split the dataset into 150 keywords for in-vocabulary training and the remaining keywords for zero-shot out-of-vocabulary testing. We chose 177 samples from 28 in-vocabulary words and all the 294 samples from out-of-vocabulary words for testing. All Tamil language keyword scripts are transliterated to English such that keywords sound as similar as possible to ASR's English character vocabulary. Though few phonemes in the Tamil language don't exist in English and several Tamil phonemes sound differently, these characters are converted to the closest sounding English characters. We also share our transliteration of Tamil keywords here[1]

4.2. Training

HyperParameters for TIMIT training: We used the OpenSeq2Seq [25] DeepSpeech2 pretrained model for our proposed KWS model. The raw signal is applied with data augmentation operations such as adding noise and speed perturbation. The model is trained with a polynomial decay learning rate with power 0.5 and uses an initial learning rate of $1 \times e^{-3}$. In addition, an L2 penalty of $1 \times e^{-5}$ is used. We use a batch size of 32 and the Adam optimizer [28]. We train the model up to 500 epochs and select the last model to report the accuracy on the test set. The value of λ_1 is 1 and λ_2 is 20. We keep the hyperparameters constant for all the experiments in this work. We also train a few other models: with CTC loss and triplet loss separately after loading the pretrained weights, with CTC and triplet loss together without loading pretrained weights. We also measure the performance of the pretrained model without any training.

Triplet Mining: The triplets for the triplet loss are mined online for each batch rather than mining it beforehand. A batch hard strategy is used where the distance between all the pairs in a batch is calculated, and then for each reference sample, the positive sample which lies farthest, and the negative sample which lies closest, are considered for calculating the triplet loss. Each batch has more than one sample for each keyword to help learn the similarity between the positive pairs, and each batch has more than one keyword, with each keyword on default having an equal number of samples.

Baseline Training: We compare our work to an Acoustic Word Embeddings system implemented by [11]. In their work, they train two LSTM networks; one receives an audio signal, and the other receives a sequence of letters corresponding to a term that has or has not been uttered within the audio signal. Their goal is to bring the LSTM outputs to produce embedding vectors. They use a contrastive loss function. The distance between the LSTMs' outputs should be smaller if the input term has been uttered in the audio signal than if the term has not been uttered in the audio signal. We train their Multi-View code for 1000 epochs with their best performing objective for baseline comparison.

Low resource Language Training: The model is trained on MSWC Tamil split for 1000 epochs on the training set. We train three models: with only triplet loss, triplet and CTC loss with pretrained weights, and without pretrained weights. The performance is measured on in-vocabulary and zero-shot out-of-vocabulary words from the test split from the same dataset. The CTC loss is applied to the transliterated text during training.

Fixed Vocabulary Training: We compare our model's performance for classification tasks on the Speech Commands dataset using kNN. We train our architecture with this dataset and obtain the embeddings for the test set that are later classified using the kNN algorithm. We train our model for 300 epochs for this task. We use [19] to compare the results. They have used a triplet loss based embedding trained on same dataset and used a variant of kNN to show accuracy performance.

We perform additional end-to-end training with the speech commands dataset. We add additional training layers to do end-to-end training. We attach a fully connected layer to the DeepSpeech2 backbone to reduce the model dimension from 1600 to 35 classes. We then add a softmax layer to the fully connected layer to produce probabilities for each class. The model is trained with a cross-entropy loss function for 300 epochs. We observe the top-1 accuracy results from softmax probability outputs for this experiment. We perform the fixed vocabulary training to compare our method's performance against the fixed vocabulary keyword classification task.

4.3. Results

TIMIT evaluation: For each pair of words from the test data set, the word discrimination system decides whether the pair contains the same or different words. The system response labels and actual ground truth labels were used to estimate the average precision (AP) as an overall system performance measure. To determine the AP value, for each pair of words from the test set, the distance between acoustic embeddings of both words was calculated, and a threshold was applied to determine if the pair represents the same or different words. The total number of pairs in the test set was 7.9 million for the TIMIT dataset. This includes both in-vocabulary and out-of-vocabulary words. By sweeping the threshold value, a precision-recall curve was ob-

[1] https://github.com/Kirandevraj/GeneralizedKWS

Table 1: *Average Precision evaluation for Generalized Keyword Spotting model and Multi-View model on TIMIT dataset for ALL(both IV and OOV), in-vocabulary(IV) and zero-shot out-of-vocabulary(OOV) test keywords samples.*

Model	ALL	IV	OOV
Multi-View [11]	0.328	0.528	0.218
Pretrained ASR	0.848	0.911	0.803
Pretrained + CTC	0.903	0.954	0.750
Pretrained + Triplet	0.936	0.975	0.700
CTC + Triplet	0.965	0.990	0.800
Pretrained + CTC + Triplet	**0.971**	**0.991**	**0.843**

Table 2: *Average Precision evaluation for Generalized Keyword Spotting Model for Low Resource Language trained on Tamil split of MSWC dataset. The columns indicate ALL(both IV and OOV), in-vocabulary(IV) and out-of-vocabulary(OOV) test keywords samples.*

Model	ALL	IV	OOV
Pretrained + Triplet	0.030	0.061	0.057
CTC + Triplet	0.161	0.300	0.253
Pretrained + CTC + Triplet	**0.295**	**0.587**	**0.321**

tained from which the AP was estimated. We calculate the AP value for all the trained models on the TIMIT dataset and report the results in Table 1. It can be observed that the performance of the model that was trained with loading pretrained weights and CTC and triplet loss has significant performance improvement over Multi-View and other models. We also observed that the model with Triplet + CTC loss was able to converge relatively faster than the model with only triplet loss.

The performance of in-vocabulary and out-of-vocabulary words is calculated separately for both Multi-View and our model and is reported in Table1. The total number of pairs used to calculate AP in in-vocabulary is 1.7 million, and out-of-vocabulary is 2.2 million.

Low Resource Language Evaluation: We calculate the AP value for the generalized keyword spotting models for low resource language on the Tamil MSWC test dataset, and the results are reported in Table 2. It can be observed that pretraining and fine-tuning with CTC and triplet loss has helped the model in achieving better performance in the Tamil language. Although pretraining and fine-tuning with only triplet loss has not helped in generalization. The total number of pairs in the test set was 110K. The total number of pairs used to calculate AP in in-vocabulary is 15K, and out-of-vocab is 43K. We hereby share this as the initial benchmark for low-resource zero-shot out of vocabulary keyword spotting.

Speech Commands Evaluation: We obtain all the embeddings for the train and test set from our model trained on Google Speech Commands dataset V2 (35 classes) with triplet and CTC loss. We then apply kNN to classify the test set to calculate accuracy. We compare our results with [19, 20, 29]. We achieve a classification accuracy of 96.2%. We have tested kNN for several values of k and have found that for the speech commands dataset, the best performing value for k is 7. For the KWS application, larger datasets occupy a lot of memory for the kNN part of the model. Our approach has the potential to achieve competitive results on classification tasks through kNN for small keyword sets. The end-to-end training on the same DeepSpeech2 backbone with Fully Connected and Softmax layer achieves 94.4%. Our method performs slightly better than end-to-end training on the same backbone showing the method is able to generalize to strong cross-entropy loss-based methods. The results are shown in Table 3.

Table 3: *Accuracy evaluation of our model on classification task trained on Google Speech Commands dataset V2. The number of target classes is* 35*. We use our Pretrained+CTC+Triplet embeddings with kNN for accuracy comparison. We additionally show the performance of end-to-end training on our DeepSpeech2 method.*

Model	Accuracy
Pretrained ASR	85.7
Attention RNN [20]	93.9
AST [29]	98.1
Res15 [19]	97.0
Pretrained + FC-Softmax	94.4
Pretrained + CTC + Triplet	96.2

Table 4: *Keywords retrieved by the Pretrined ASR vs Pretrained + triplet vs Our Pretrained + CTC + Triplet model on OOV and IV keywords. Bold represents query and the closest sample belonging to the same keyword.*

Model	Query: Top 3 unique nearest neighbours
Pretrained	*magnetic*: economic, barometric, diagram
Pre + triplet	*magnetic*(oov): **magnetic**, money, nobody
Ours	*magnetic*(oov): **magnetic**, money, barometric
Pretrained	*livestock*: **livestock**, stopwatch, stockings
Pre + triplet	*livestock*(oov): muscular, catastrophic, extra
Ours	*livestock*(oov): **livestock**, extra, electron
Pretrained	*getting*: looking, would, meeting
Pre + triplet	*getting*(iv): began, **getting**, bedroom
Ours	*getting*(iv): **getting**, heating, eating
Pretrained	*program*: **program**, arriving, problem
Pre + triplet	*program*(iv): causeway, cranberry, corduroy
Ours	*program*(iv): **program**, problem, popular

4.4. Analysis

We show the nearest neighbors for four sample queries in the test set in Table 4. Our model performs better than the pretrained model for the keyword *magnetic* and the keyword *livestock* even though it is unseen during training even while the triplet model has missed *livestock*. The model improves the representation for keyword *getting* from the pretrained model after seeing it. The model preserves the representation for the keyword *program* while the triplet model fails to retrieve it. It can be observed that the keywords that lie closer have similar phonemes for our model in all the cases in Table 4.

5. Conclusion

A novel method of learning acoustic word embeddings by transferring pretrained ASR architecture for KWS with triplet network and CTC is proposed and experimented. We show this method generalizes to zero-shot out of vocabulary keywords. It is demonstrated that the proposed model is competitive with recent deep learning benchmarks for word discrimination and classification tasks. We show initial zero-shot generalization results for low-resource languages and share the text transliteration for these keywords. It is also observed that pretraining in one language helps in KWS on a different low-resource target language.

6. References

[1] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*. PMLR, 2016, pp. 173–182.

[2] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE ICASSP*. IEEE, 2015, pp. 5206–5210.

[3] M. Mazumder, C. R. Banbury, J. Meyer, P. Warden, and V. J. Reddi, "Few-shot keyword spotting in any language," in *Interspeech*, 2021.

[4] T. S. Fuchs, Y. Segal, and J. Keshet, "Cnn-based spoken term detection and localization without dynamic programming," *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6853–6857, 2021.

[5] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, p. 27403, 1993.

[6] M. Mazumder, S. Chitlangia, C. Banbury, Y. Kang, J. M. Ciro, K. Achorn, D. Galvez, M. Sabini, P. Mattson, D. Kanter *et al.*, "Multilingual spoken words corpus," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[7] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[8] T. Fuchs and J. Keshet, "Spoken term detection automatically adjusted for a given threshold," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1310–1317, 2017.

[9] B. Yusuf and M. Saraclar, "An empirical evaluation of dtw subsampling methods for keyword search." in *INTERSPEECH*, 2019.

[10] D. Ram, L. Miculicich, and H. Bourlard, "Multilingual bottleneck features for query by example spoken term detection," in *2019 IEEE ASRU*, 2019, pp. 621–628.

[11] W. He, W. Wang, and K. Livescu, "Multi-view recurrent neural acoustic word embeddings," in *Proc. ICLR*, 2017.

[12] D. Ram, L. M. Werlen, and H. Bourlard, "Cnn based query by example spoken term detection." in *Interspeech*, 2018, pp. 92–96.

[13] H. Kamper, W. Wang, and K. Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," in *2016 IEEE ICASSP*. IEEE, 2016, pp. 4950–4954.

[14] D. Shitov, E. Pirogova, T. A. Wysocki, and M. Lech, "Learning acoustic word embeddings with dynamic time warping triplet networks," *IEEE Access*, vol. 8, pp. 103 327–103 338, 2020.

[15] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, "End-to-end asr-free keyword search from speech," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1351–1359, 2017.

[16] A. Rosenberg, K. Audhkhasi, A. Sethy, B. Ramabhadran, and M. Picheny, "End-to-end speech recognition and keyword search on low-resource languages," in *2017 ICASSP*. IEEE, 2017, pp. 5280–5284.

[17] H. Zhang, J. Zhang, and Y. Wang, "Sequence-to-sequence models for small-footprint keyword spotting," *arXiv preprint arXiv:1811.00348*, 2018.

[18] Y. Zhuang, X. Chang, Y. Qian, and K. Yu, "Unrestricted vocabulary keyword spotting using lstm-ctc." in *Interspeech*, 2016, pp. 938–942.

[19] R. Vygon and N. Mikhaylovskiy, "Learning efficient representations for keyword spotting with triplet loss," in *SPECOM*, 2021.

[20] D. C. de Andrade, S. Leo, M. Viana, and C. Bernkopf, "A neural attention model for speech command recognition," *ArXiv*, vol. abs/1808.08929, 2018.

[21] T. Kim and J. Nam, "Temporal feedback convolutional recurrent neural networks for keyword spotting," *ArXiv*, vol. abs/1911.01803, 2019.

[22] C. T. Lengerich and A. Y. Hannun, "An end-to-end architecture for keyword spotting and voice activity detection," *ArXiv*, vol. abs/1611.09405, 2016.

[23] Z. Zhu, Z. Wu, R. Li, H. M. Meng, and L. Cai, "Siamese recurrent auto-encoder representation for query-by-example spoken term detection," in *INTERSPEECH*, 2018.

[24] W. He, W. Wang, and K. Livescu, "Multi-view recurrent neural acoustic word embeddings," *arXiv preprint arXiv:1611.04496*, 2016.

[25] O. Kuchaiev, B. Ginsburg, I. Gitman, V. Lavrukhin, J. Li, H. Nguyen, C. Case, and P. Micikevicius, "Mixed-precision training for nlp and speech recognition with openseq2seq," *arXiv preprint arXiv:1805.10387*, 2018.

[26] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International workshop on similarity-based pattern recognition*. Springer, 2015, pp. 84–92.

[27] B. Kim, M. Lee, J. Lee, Y. Kim, and K. Hwang, "Query-by-example on-device keyword spotting," *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 532–538, 2019.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] Y. Gong, Y.-A. Chung, and J. R. Glass, "Ast: Audio spectrogram transformer," *ArXiv*, vol. abs/2104.01778, 2021.