Learning Efficient Representations for Keyword Spotting with Triplet Loss

Roman Vygon
Higher IT School,
Tomsk State University & NTR Labs
Tomsk, Russia
rvygon@ntr.ai

Nikolay Mikhaylovskiy
Higher IT School,
Tomsk State University & NTR Labs
Moscow, Russia
nickm@ntr.ai

***Abstract*—In the past few years, triplet loss-based metric embeddings have become a de-facto standard for several important computer vision problems, most notably, person reidentification. On the other hand, in the area of speech recognition the metric embeddings generated by the triplet loss are rarely used even for classification problems. We fill this gap showing that a combination of two representation learning techniques: a triplet loss-based embedding and a variant of kNN for classification instead of cross-entropy loss significantly (by 26% to 38%) improves the classification accuracy for convolutional networks on a LibriSpeech-derived LibriWords datasets. To do so, we propose a novel phonetic similarity based triplet mining approach. We also improve the current best published SOTA (for small-footprint models) for Google Speech Commands dataset V2 10+2-class classification by about 16%, achieving 98.37% accuracy, and the current best published SOTA for 35-class classification on Google Speech Commands dataset V2 by 47%, achieving 97.0% accuracy.[1]**

Keywords— keyword spotting, spoken term detection, triplet loss, kNN, representation learning, audio classification

I. Introduction

The goal of keyword spotting is to detect a relatively small set of predefined keywords in a stream of user utterances, usually in the context of small-footprint device [1]. Keyword spotting (KWS for short) is a critical component for enabling speech-based user interactions for such devices [2]. It is also important from an engineering perspective for a wide range of applications [3].

In this article we show how the use of the triplet loss-based embeddings allows us to improve the classification accuracy of the existing small-footprint neural network architectures.

A. Previous work on KWS

The first work on KWS was most likely published in 1967 [4]. Over years, a number of machine learning architectures for small-footprint KWS have been proposed (see, for example [5][6][7][8][9]. With the renaissance of neural networks, they become the architecture class of choice for KWS systems (see, for example, [1][2][10][11][12][13][14]). Probably, the only – but notable – exception from this trend is the very recent work of Lei et al. [15] that uses Tsetlin machines for keyword spotting for their extremely low power consumption.

Publication of the Google Speech Command dataset [16] have provided a common ground for KWS system evaluation and allowed for accelerating research. Further, we denote V1 and V2 versions 1 and 2 of the dataset, respectively. When publishing the dataset, Warden [16] have also provided a baseline model based on the convolutional architecture of Sainath and Parada [11], achieving the accuracy of 85.4% and 88.2% on V1 and V2, respectively. The respective Kaggle competition winner has achieved 91% accuracy on V1.

Since the publication of the Google Speech Command dataset led to a vast corpus of work appearing in the past three years, we will only briefly discuss the most relevant recent work. Jansson [17] suggested an interesting fully-convolutional model working out of raw waveforms, but, probably, a bit ahead of time and did not improve on previous accuracy results. de Andrade et al. [3] have proposed an attention-based recurrent network architecture and achieved the SOTA on 2, 10, 20-word and full-scale versions of the dataset. Majumdar and Ginsburg [18] have published a lightweight separable convolution residual network architecture MatchboxNet, achieving the new SOTA of 97.48% on V1 and 97.63% on V2. Mordido et al. [19] have suggested an interesting improvement to MatchboxNet model, replacing 1x1-convolutions in 1D time-channel separable convolutions by constant, sparse random ternary matrices with weights in {-1; 0; +1}.

Rybakov, Kononenko et al. [20] tested many of the existing models and proposed a multihead attention-based recurrent

[1] Code is available at https://github.com/roman-vygon/triplet_loss_kws

neural network architecture, achieving a new SOTA of 98% on V2. Wei et al. [21] proposed a new architecture, EdgeCRNN, which is based on depthwise separable convolution and residual structure, apparently drawing inspiration from MatchboxNet [18] and Attention RNN [3], to achieve a slight improvement in accuracy and a SOTA of 98.05%. Tang et al. [22] have released Howl - a productionalized, open-source wake word detection toolkit, explored a number of models and achieved nearly-SOTA accuracy with a residual convolutional network architecture.

Berg et al. have published a work [23] on gigantic, by KWS measures, models. Their smallest model is about 5 times larger than ours, mid-sized one, with a comparable performance, 20 times larger, and the model achieving the state-of-the-art of 98.6% is about 50 times larger than ours. Given that we operate in a small-footprint setting, these models are no competitors to the other approaches listed above and ours as well.

B. *Previous work on the use of triplet loss for the metric embedding learning*

The goal of metric embedding learning is to learn a function $f: R^F \rightarrow R^D$, which maps semantically similar points from the data manifold in R^F onto metrically close points in R^D, and semantically different points in R^F onto metrically distant points in R^D [24].

The triplet loss for this problem was most likely first introduced in [25] in the framework of image ranking:

$$l(p_i, p_i^+, p_i^-) = \{0, g + D(f(p_i), f(p_i^+)) - D(f(p_i), f(p_i^-))\} \quad (1)$$

where p_i, p_i^+, p_i^- are the anchor image, positive image, and negative image, respectively, g is a gap parameter that regularizes the gap between the distance of the two image pairs: (p_i, p_i^+) and (p_i, p_i^-), and D is a distance function that can be, for example, Euclidean distance in the image embedding space:

$$D(f(P), f(Q)) = \|f(P) - f(Q)\|_2^2 \quad (2)$$

A similar loss function was earlier proposed by Chechik et al. in [26], but the real traction came to the triplet loss in the area of face re-identification after the works of Schroff, Kalenichenko, and Philbin on FaceNet [27] and Hermans, Beyer, and Leibe [24].

In the speech domain, the use of triplet loss is more limited, but there still are several important works we need to mention. In particular, Huang J. et al [28], Ren et al. [29], Kumar et al. [30], and Harvill et al. [31] use triplet loss with varied neural network architectures for the task of the speech emotion recognition. Bredin [32] and Song et al. [33] use triplet-loss based learning approaches for the speaker diarization, and Zhang and Koshida [34] and Li et al. [35] – for the related task of speaker verification.

Turpault et al. [36] propose a strategy for augmenting data with transformed samples, in line with more recent works in varied machine learning areas.

The most similar works to ours are probably [37], [38], [39], [40], and [41], but there are many important differences:

- Sacchi et al. [37] operate in the open-vocabulary setting, which required the authors to design a system with a common embedding for text and speech, while we concentrate on improving the quality of existing low-footprint architectures for closed-vocabulary keyword spotting
- Shor et al. [38] concentrate on building an unified embedding that works well for non-semantic tasks, while we concentrate on the semantic task of keyword spotting
- Yuan et al. [39] operate in a two-stage detection/ classification framework and use a BLSTM network with a mix of triplet, reverse triplet and hinge loss
- Huh et al. [40] start from the same res15 model as we do, but primarily focus on detection metrics and use SVM for classification, so our classification metrics are significantly better
- Huang et. al [41] concentrate on Query-by-Example KWS application and adopt the softtriple loss - a combination of triplet loss and softmax loss

C. *Our contributions*

Our contributions in this work are the following:

- We show that combining two representation learning methods: triplet-loss based metric embeddings and a kNN classifier allows us to significantly improve the accuracy of CNN-based models that use cross-entropy to classify audio information
- We propose a novel batch sampling approach based on phonetic similarity that allows to improve F1 metric when classifying highly imbalanced datasets

II. MODEL ARCHITECTURES

Most of the current state-of-the-art keyword spotting architectures are present in the work of Rybakov et al. [20], with the best model to date being the Bidirectional GRU-based Multihead attention RNN. It takes a mel-scale spectrogram and convolves it with a set of 2D convolutions. Then two bidirectional GRU layers are used to capture two-way long term dependencies in the audio data. The feature in the center of the bidirectional LSTM's output sequence is projected using a dense layer and is used as a query vector for the multi-head attention (4 heads) mechanism. Finally, the weighted (by attention score) average of the bidirectional GRU output is processed by a set of fully connected layers for classification.

We have mostly experimented with ResNet-based models res8 [22] [1] and res15 [42] [1]. The initial experiments have shown that RNN-based architectures show significantly worse results when trained for the triplet loss, so they were discarded in our later work We used the encoder part of each of the models above to generate triplet-loss based embeddings, that are later classified using the K-Nearest Neighbor (kNN) algorithm.

A. *Input Preprocessing*

Sixty-four-dimensional Mel-Spectrograms are constructed and stacked using a 25-millisecond window size and a 10-

millisecond frame shift. Our implementation stacks all such shifted 30-millisecond windows within the one-second sample of Google Speech Commands. Libriwords samples are constrained to have a duration of 0.1-3 seconds.

B. *Resnet architecture.*

Our resnet implementation is taken directly from [42] with very minor code changes and is depicted in Figure 1.

When working with triplet loss, the softmax layer is removed.

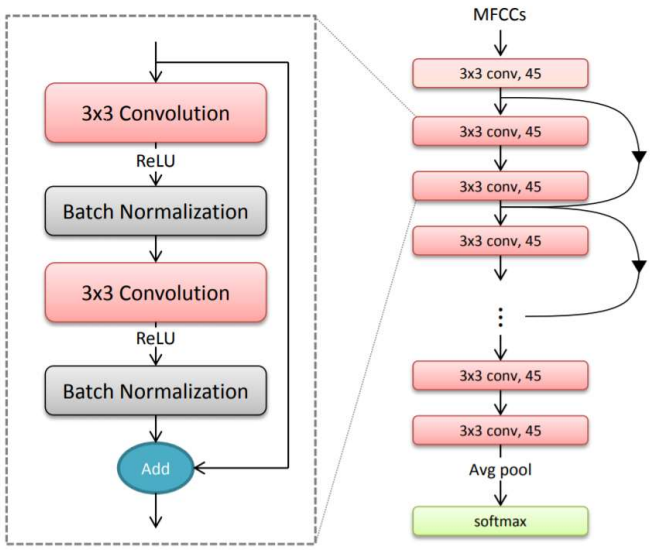

Figure 1: res architecture (from [1])*

TABLE I. ENCODER MODEL SIZES FOR THE KEY MODELS STUDIED.

	Embedding dimension	**Model encoder size, [K]**
Mh-Att-RNN	256	743
res8	128	885
res15	45	109
Att-RNN	128	202

TABLE I. above compares the model sizes for the main models studied.

III. EXPERIMENTS

A. *Datasets and tasks*

1) SpeechCommands

Google Speech Commands dataset Version 1 has 65K utterances from various speakers, each utterance 1 second long. Each of these utterances belongs to one of 30 classes corresponding to common words like ”Go”, ”Stop”, ”Left”, ”Down”, etc. Version 2 has 105K utterances, each 1 second long, belonging to one of 35 classes. The sampling rate of both datasets is 16kHz.

In our experiments we have considered the following tasks based on these datasets [3][16]:

- Recognition of all 35 words using Google Speech Dataset V2
- Recognition of 10 words ("Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", and "Go") and additional labels for “Unknown” and “Silence”.

For these tasks and each architecture studied we have measured top-1 classification accuracy.

2) LibriWords Datasets

To further explore the possibilities of triplet loss models we needed a dataset that consists of a large number of different words to classify.

Thus, we have used LibriSpeech [43] - a collection of 1,000 hours of read English speech. The dataset was split on the word level by Lugosch et al. [44]. Since LibriSpeech is aligned on sentence level only, the Montreal Forced Aligner [45] was used to obtain intervals for individual words. The alignments are available online [44]. Further we call this derived dataset LibriWords.

We have created four different versions of the dataset (LibriWords10, LibriWords100, LibriWords1000, LibriWords10000) that correspond to the first 10, 100 etc. words by popularity in the LibriSpeech 1000h corpus. For example, the LibriWords10 words are: “the”, “and”, “of”, “to”, “a”, “in”, “he”, “I”, “that”, “was”.

Durations of the words range from 0.03 seconds to 2.8 seconds, with mean duration of 0.28 seconds. The details on the datasets metrics are available in the Appendix 1. We have split the dataset into train\val\test in in 8:1:1 proportion, and tried to make sure this proportion holds for each word in the dataset. We release NeMo-like manifests for ease of use and reproduction. Since the motivation behind the dataset is to model real-life speech recognition scenarios, there was no further quality assurance on the data.

B. *Approach to training models*

1) Batch sampling

When working with Speech Commands and LibriWords10 datasets, to ensure a meaningful representation of the anchor-positive distances, following [27], we sample an equal number of objects from all the classes available. For unbalanced datasets with a large number of words, we also needed an efficient class-sampling method, otherwise the network will often train on irrelevant batches where embeddings of the words are already far from each other. To achieve better class selection we have used three sampling approaches:

- Uniform: sample *batch_size* classes randomly from a uniform distribution.
- Proportional: sample *batch_size* classes randomly from a distribution proportional to the word distribution in the dataset. Motivation behind this approach is twofold. First, the popular words are short (the, a, I)) so they are not easy to distinguish from the rest. Second, if you equally train on them, there will be the same amount of errors, and that’s a lot in terms of the absolute value. (If we classify 2% of a popular word incorrectly, this would significantly spoil the metric for the entire dataset).

- Phonetic: Calculate a matrix of phonetic similarity for all the words in the dataset, sample batch_size/2 classes, then, for each sampled class add a random phonetically similar word to the batch. Similarity score is calculated using SoundEx, Caverphone, Metaphone and NYSIIS algorithms [47].

These sound similarity algorithms were all developed with tasks different from ours in mind. To achieve a baseline applicable to LibriWords we used a weighted average of distances calculated using all 4 algorithms. The weights go as follows:

$$D_{Phonetic} = D_{Soundex} * 0.2 + D_{caverphon} * 0.2 + D_{metaphone} * 0.5 + D_{nysiis} * 0.1$$

The metaphone algorithm has a bigger weight due to its original task being the nearest to ours. The optimal use of these algorithms is a matter of future research, for example, we had to adjust manually the distances of a handful of pairs of words: e.g. the pair "know-no" had a large distance while being similar. The problem was found while analyzing the confusion matrix.

We have evaluated these three triplet mining approaches alone and in combinations, mixing them with equal probabilities. Thus, for example, Uniform+Phounetic in the TABLE II. below means that we have sampled 50% of triplets in the batch using Uniform strategy, and 50% using Phonetic strategy, and Uniform+Proportional +Phonetic means that 1/3 of the triplets were sampled using Uniform strategy, 1/3 - using Phonetic strategy, and 1/3 - using Proportional strategy.

The results in the TABLE II. show that the proportional sampling method improves the accuracy by increasing the score of more popular words while the phonetic sampling method improves the F1 metric due to better classification of difficult pairs like "at"-"ate", "an"-"anne". Uniform sampling usage is essential as one of the sampling strategies, as it provides the proper class coverage.

TABLE II. THE EFFECTS OF THE DIFFERENT SAMPLING STRATEGIES FOR TRIPLET LOSS OF RES15 MODEL ON LIBRIWORDS10000

Method(s)	Accuracy	F1
Uniform	79.4	0.72
Proportional	77.1	0.61
Phonetic	76.9	0.73
Uniform+Phonetic	78.9	**0.76**
Uniform+Proportional	**81.2**	0.74
Proportional+Phonetic	80.0	0.72
Uniform+Proportional +Phonetic	80.8	0.75

2) Triplet selection

An important part of TL models is the selection of triplets used to calculate the loss, since taking all possible triplets from a batch is computationally expensive. We have used a randomized approach to the online batch triplet mining based on [24], where the negative sample to a hard pair of the anchor and a positive sample is selected randomly from the set of negative samples resulting in non-zero loss. Our initial experiments have shown that this modification of the online batch triplet mining performs better than hard or semi-hard batch loss options.

3) Optimization and training process

Baseline models were trained until they reached a plateau on a validation set. We monitored the validation accuracy of triplet loss models each 1k batches and stopped the training process if the accuracy didn't increase for more than .1% for 3 consecutive times. The number of epochs is listed in the TABLE III. below.

Three augmentation techniques were used:

1. Shifting samples in range (-100ms; +100ms).
2. SpecAugment.
3. Adding background noise from audio files in Google Speech Commands Dataset.

The decrease in epochs for larger datasets is due to class-imbalance – triplet models sample classes directly, so instead of seeing all objects in the dataset it sees the same number of objects, but distributed more evenly between classes. The baseline, cross-validation based models converge to predict the most popular words well, while ignoring the rest. One can see this from the low F1 metric on LibriWords10000 dataset. The batch size was 35*10 for TL-res8, 35*4 for TL-res15 and 128 for the baseline models.

Training was done using the Novograd [48] algorithm with initial learning rate of 0.001 and cosine annealing decay to 1e-4.

4) Influence of kNN

We have tested kNN for several values of k, and have found that for LibriWords the best performing value varies depending on the dataset size, while for Speech Commands the best performing value was *k=5* (see 0).

TABLE III. THE NUMBER OF EPOCHS MODELS WERE TRAINED FOR

	TL, epochs	Baseline, epochs
Speech Commands	30	30
Libri10	10	30
Libri100	5	10
Libri1000	5	7
Libri10000	3	5

As the model size is of a great concern for the keyword spotting application, and for the larger datasets kNN part of the model can take a lot of memory, we have also studied the effect of kNN quantization available from [49] on the size, speed and accuracy of the resulting model, varying the number of segments for the Product Quantizer.

TABLE IV. CLASSIFICATION ACCURACY FOR RES15 MODEL TRIPLET LOSS EMBEDDINGS WITH KNN CLASSIFICATION FOR VARIOUS K.

k	1	5	10	30
Speech Commands V2 / 12	98.18	**98.37**	98.27	98.29
LW10	89.91	91.48	**91.74**	91.72
LW100	83.93	86.53	86.9	**86.98**
LW1000	80.43	83.82	84.29	**84.37**
LW10000	77.57	80.82	**81.17**	80.62

For each dataset/task there is an optimal number of segments that reduces accuracy by 1.6% - 13.6%, and reduces the memory consumption by a factor of 7 to 13 (see Tables VIII-XIII, Appendix 2)

IV. RESULTS AND DISCUSSION

The results below were obtained by training a model for 3 different runs in each scenario and averaging the results to avoid the "lucky seed" effect. We can see that triplet loss + kNN based models provide better accuracy than baseline ones, achieve or match state of the art results on Speech Commands dataset, while being more lightweight and faster in convergence than the mh-att-rnn [20] model.

In particular, triplet loss + kNN based models improve the accuracy on the datasets studied by 14% to 38% and F1 measure by 8% to 57% compared to extremely strong crossentropy based baselines (see TABLE V.) The bigger the number of classes in the dataset, the bigger the difference between crossentropy and triplet loss based classifiers.

On Google Speech Commands dataset V2/35 task, our res15 network trained with triplet loss and kNN classifier, achieves state of the art, improving the best previously published result [3] by 47%. On Google Speech Commands dataset V2/12 task it improves the state of the art [20] by about 16% (see TABLE VI.)

TABLE V. COMPARISON OF ACCURACY AND F1 MEASURE OF TRIPLET LOSS AND CROSSENTROPY LOSS BASED RES15 MODELS

Task	Triplet Loss		Crossentropy		Relative improvement	
	Accuracy, %	*F1*	*Accuracy, %*	*F1*	*Accuracy, %*	*F1,%*
Speech Commands V2 35	97.0	0.965	95.96	0.955	25.74	22.22
Speech Commands V2 12	98.37	0.98	97.8	0.963	25.9	45.94
LibriWords10	91.7	0.90	88.8	0.88	26.25	16.67
LibriWords100	86.9	0.87	82.3	0.81	25.99	31.58
LibriWords 1000	84.3	0.86	78.2	0.78	27.94	36.36
LibriWords 10000	81.2	0.75	69.3	0.41	38.66	57.63

TABLE VI. MODEL ACCURACY COMPARISON ON GOOGLE SPEECH COMMANDS DATASET TASKS

Model	Loss	Model Size, KB	V2 35 accuracy	V2 12 accuracy
res8	Triplet	901	95.33	97.48
	Crossentropy	885	95.25	97.39
res15	Triplet	252	**97.0**	**98.37**
	Crossentropy	237	95.96	97.8
EdgeCRNN [21]	Crossentropy			98.05
Mh-Att-RNN [20]	Crossentropy	743		98.0
Attention RNN [3]	Crossentropy	202	93.9	

V. ACKNOWLEDGMENTS

The authors are grateful to

- colleagues at NTR Labs Machine Learning Research group for the discussions and support;
- Prof. Sergey Orlov and Prof. Oleg Zmeev for the computing facilities provided;
- Nikolay Shmyrev for pointing out to the works [39], [40].

VI. REFERENCES

[1] Tang, R., and J. Lin, "Deep residual learning for small-footprint keyword spotting." ArXiv 1710.10361.

[2] Zhang, Y., N. Suda, L. Lai, and V. Chandra, "Hello Edge: Keyword Spotting on Microcontrollers," arXiv 1711.07128.

[3] de Andrade, D., L. Sabato, M. Viana, and C. Bernkopf, "A neural attention model for speech command recognition." arXiv 1808.08929

[4] Teacher, C., Kellett, y., and Focht L., "Experimental, limited vocabulary, speech recognizer." IEEE Transactions on Audio and Electroacoustics, 15(3):127–130, 1967.

[5] Rohlicek, J.R., Russell, W., Roukos, S., Gish, H., "Continuous hidden Markov modeling for speaker-independent word spotting." In: Acoustics, Speech, and Signal Processing 1989, pp. 627–630. IEEE

[6] Szoke I., Schwarz P., Matejka P., et al., "Phoneme based acoustics keyword spotting in informal continuous speech" In: Matousek V., Mautner P., Pavelka T. Text, Speech and Dialogue, Berlin: Springer-Verlag, 2005:302-309 doi: 10.1007/11551874_39.

[7] Zhang, S., Shuang, Z., Shi, Q., Qin, Y., "Improved mandarin keyword spotting using confusion garbage model." In: 2010 20th International Conference on Pattern Recognition (ICPR), pp. 3700–3703.

[8] Greibus M., Telksnys L., "Speech Keyword Spotting with Rule Based Segmentation." In: Skersys T., Butleris R., Butkiene R. (eds) Information and Software Technologies. ICIST 2013. Communications in Computer and Information Science, vol 403. Springer, Berlin, Heidelberg.

[9] Principi, S. Squartini, R. Bonfigli, G. Ferroni, and F. Piazza, "An integrated system for voice command recognition and emergency detection based on audio signals," Expert Syst. Appl., vol. 42, no. 13, pp. 5668–5683, Aug. 2015, doi: 10.1016/j.eswa.2015.02.036.

[10] Chen G., Parada C., and Heigold G., "Small-footprint keyword spotting using deep neural networks." In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, pages 4087–4091.

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

[12] Arik S. O., Kliegl M., Child R., Hestness J., Gibiansky A., Fougner C., Prenger R., and Coates A., "Convolutional recurrent neural networks for small-footprint keyword spotting." arXiv:1703.05390, 2017.

[13] Sun M., Raju A., Tucker G., Panchapagesan S., Fu G., Mandal A., Matsoukas S., Strom T., and Vitaladevuni S., "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting." In Spoken Language Technology Workshop (SLT), 2016 IEEE, pages 474–480. IEEE, 2016.

[14] He, Y., Prabhavalkar, R., Rao, K., Li, W., Bakhtin, A., and McGraw, I., "Streaming Small-Footprint Keyword Spotting using Sequence-to-Sequence Models." arXiv:1710.09617

[15] J. Lei *et al.*, "Low-Power Audio Keyword Spotting using Tsetlin Machines," pp. 1–20, 2021, [Online]. Available: http://arxiv.org/abs/2101.11336.

[16] Warden P., "Speech commands: A public dataset for single-word speech recognition."arXiv:1804.03209.

[17] Jansson, P., "Single-word speech recognition with Convolutional Neural Networks on raw waveforms". Degree Thesis, Information technology, ARCADA University, Finland

[18] Majumdar S., and Ginsburg B., "MatchboxNet: 1D Time-Channel Separable Convolutional Neural Network Architecture for Speech Commands Recognition" arXiv:2004.08531

[19] G. Mordido, M. Van Keirsbilck, and A. Keller, "Compressing 1D Time-Channel Separable Convolutions using Sparse Random Ternary Matrices," 2021, [Online]. Available: http://arxiv.org/abs/2103.17142.

[20] Rybakov O., Kononenko N., Subrahmanya N., Visontai M., and Laurenzo S., "Streaming keyword spotting on mobile devices" arXiv:2005.06720

[21] Wei, Y., Gong, Z., Yang, S., Ye, K., and Wen, Y., "EdgeCRNN: an edge-computing oriented model of acoustic feature enhancement for keyword spotting," *J. Ambient Intell. Humaniz. Comput.*, pp. 1–11, Mar. 2021

[22] Tang, R., Lee, J., Razi, A., Cambre, J., Bicking, I., Kaye, J., and Lin, J., "Howl: A Deployed, Open-Source Wake Word Detection System." arXiv:2008.09606.

[23] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword Transformer: A Self-Attention Model for Keyword Spotting," 2021, [Online]. Available: http://arxiv.org/abs/2104.00769.

[24] Hermans_A., Beyer L, and Leibe B., "In Defense of the Triplet Loss for Person Re-Identification" arXiv:1703.07737

[25] Wang J., Song Y., Leung T., Rosenberg C., Wang J., Philbin J., Chen B., and Wu Y., "Learning fine-grained image similarity with deep ranking." arXiv:1404.4661

[26] Chechik, G., Sharma, V., Shalit, U., and Bengio, S., "Large scale online learning of image similarity through ranking." The Journal of Machine Learning Research, 11:1109–1135, 2010.

[27] Schroff F., Kalenichenko D., and Philbin J., "FaceNet: A Unified Embedding for Face Recognition and Clustering." arXiv:1503.03832

[28] Huang J., Li Y., Tao J., and Lian Z., "Speech Emotion Recognition from Variable-Length Inputs with Triplet Loss Function." INTERSPEECH 2018: 3673-3677

[29] Ren M., Nie W., Liu A., Su Y., "Multi-modal Correlated Network for emotion recognition in speech", Visual Informatics, Volume 3, Issue 3, 2019, Pages 150-155

[30] Kumar P., Jain S., Raman B, Roy P.P., and Iwamura M., "End-to-end Triplet Loss based Emotion Embedding System for Speech Emotion Recognition", arXiv:2010.06200

[31] Harvill J., AbdelWahab M., Lotfian R., and Busso C., "Retrieving speech samples with similar emotional content using a Triplet loss function", ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, United Kingdom, 2019, pp. 7400-7404, doi: 10.1109/ICASSP.2019.8683273.

[32] Bredin H., "Tristounet: triplet loss for speaker turn embedding" arXiv:1609.04301

[33] Song H., Willi M., Thiagarajan J.J., Berisha V., Spanias A., "Triplet Network with Attention for Speaker Diarization" arXiv:1808.01535

[34] Zhang, C., Koishida, K., "End-to-End Text-Independent Speaker Verification with Triplet Loss on Short Utterances." Proc. Interspeech 2017, 1487-1491, DOI: 10.21437/Interspeech.2017-1608.

[35] Li C., Ma X., Jiang B., Li X., Zhang X., Liu X., Cao Y., Kannan A., Zhu Z., "Deep Speaker: an End-to-End Neural Speaker Embedding System" arXiv:1705.02304

[36] Turpault N., Serizel R., Vincent E., "Semi-supervised triplet loss based learning of ambient audio embeddings." ICASSP 2019, May 2019, Brighton, United Kingdom. hal-02025824

[37] Sacchi, N., Nanchen, A., Jaggi, M. and Cerňak, M., "Open-Vocabulary Keyword Spotting with Audio and Text Embeddings." 3362-3366. 10.21437/Interspeech.2019-1846.

[38] Shor J., Jansen A., Maor R., Lang O., Tuval O., de Chaumont Quitry F., Tagliasacchi M., Shavitt I., Emanuel D., HavivY., "Towards Learning a Universal Non-Semantic Representation of Speech" arXiv:2002.12764

[39] Y. Yuan, Z. Lv, S. Huang, and L. Xie, "Verifying Deep Keyword Spotting Detection with Acoustic Word Embeddings," 2019 IEEE Autom. Speech Recognit. Underst. Work. ASRU 2019 - Proc., no. 61571363, pp. 613–620, 2019, doi: 10.1109/ASRU46091.2019.9003781.

[40] J. Huh, M. Lee, H. Heo, S. Mun, and J. S. Chung, "Metric learning for keyword spotting," arXiv:2005.08776 2020

[41] J. Huang, W. Gharbieh, H. S. Shim, and E. Kim, "Query-by-Example Keyword Spotting system using Multi-head Attention and Softtriple Loss," 2021, Accessed: Mar. 23, 2021. [Online]. Available: http://arxiv.org/abs/2102.07061.

[42] R. Tang and J. Lin, "Honk: A PyTorch reimplementation of convolutional neural networks for keyword spotting," *arXiv*. arXiv, Oct. 17, 2017, Accessed: Jan. 02, 2021. http://arxiv.org/abs/1710.06554.

[43] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Aug. 2015, vol. 2015-August, pp. 5206–5210,

[44] Lugosch L., Ravanelli M., Ignoto P., Tomar V. S., and Bengio Y., "Speech model pre-training for end-to-end spoken language understanding," in Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2019, vol. 2019-Sept, pp. 814–818

[45] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi," Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH, vol. 2017-Aug, pp. 498–502, 2017, doi: 10.21437/Interspeech.2017-1386.

[46] https://zenodo.org/record/2619474, last retrieved on Jan 2nd, 2021

[47] A. F. Ahmed, M. A. Sherif, and A. C. N. Ngomo, "Do your resources sound similar?: On the impact of using phonetic similarity in link discovery," in K-CAP 2019 - Proceedings of the 10th International Conference on Knowledge Capture, 2019, vol. 8, no. 19, pp. 53–60, doi: 10.1145/3360901.3364426.

[48] B. Ginsburg *et al.*, "Stochastic Gradient Methods with Layer-wise Adaptive Moments for Training of Deep Networks," *arXiv*: http://arxiv.org/abs/1905.11286.

[49] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *arXiv*. arXiv:1702.08734v1 doi: 10.1109/tbdata.2019.2921572.

APPENDIX 1. WORD DISTRIBUTIONS IN LIBRIWORDS DATASETS

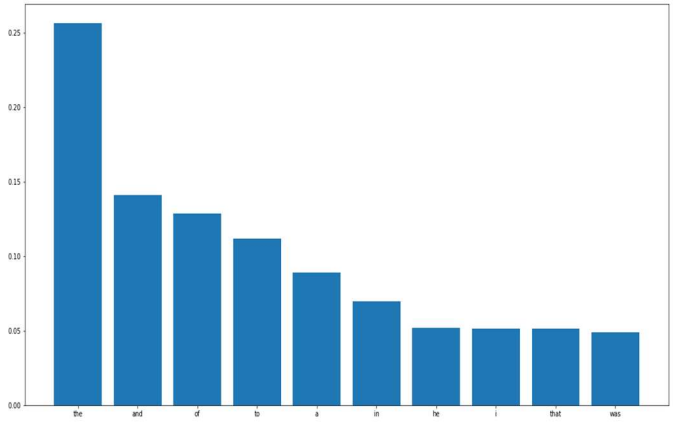

Fig. 1. Distribution of words in LibriWords10

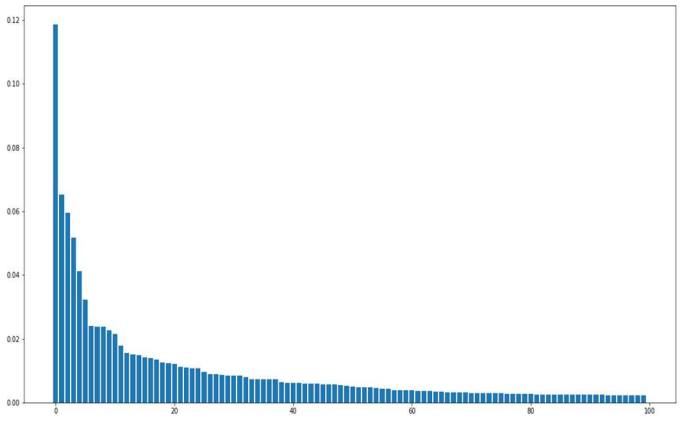

Fig. 2. Distribution of words in LibriWords100

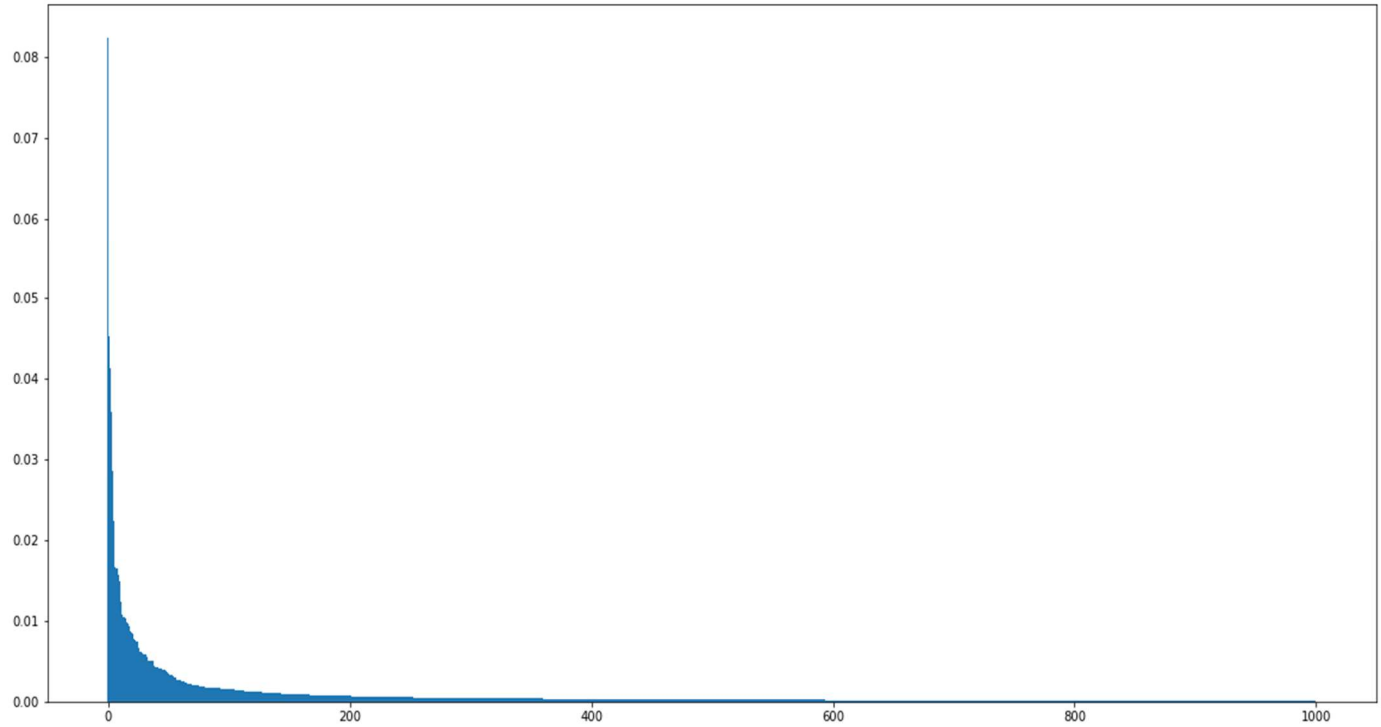

Fig. 3. Distribution of words in LibriWords1000.

TABLE VII. STATS OF LIBRIWORDS DATASETS

	Total words	**Most popular word**	**Least popular word**	**Class imbalance**
libri10	875 043	the 224 240	was 42 757	5.2
libri100	1 890 091	the 224 240	never 4 121	54.4
libri1000	2 723 023	the 224 240	path 329	681.6
libri10000	3 394 530	the 224 240	parade 21	10678

APPENDIX 2. INFLUENCE OF KNN QUANTIZATION ON SPEED, MEMORY CONSUMPTION AND ACCURACY OF CLASSIFIERS

TABLE VIII. MEMORY CONSUMPTION, TEST SET PREDICT TIME AND ACCURACY FOR DIFFERENT KNN QUANTIZATION SEGMENT NUMBER FOR LIBRIWORDS10 DATASET.

	Memory, MB	**Time, s**	**Accuracy, %**
Basic	246	17.15	**91.74**
4	5.95	0.53	91.34
8	7.87	0.62	91.48
16	11.7	0.73	91.56
32	19.4	1.1	**91.61**
64	34.8	2.33	91.58

TABLE IX. MEMORY CONSUMPTION, TEST SET PREDICT TIME AND ACCURACY FOR DIFFERENT KNN QUANTIZATION SEGMENT NUMBER FOR LIBRIWORDS100 DATASET.

Basic	638	118.6	**86.9**
4	15.1	1.72	91.34
8	20.1	2.08	85.23
16	30.1	2.7	86.11
32	50	4.18	**86.24**
64	90	8.98	86.22

TABLE X. MEMORY CONSUMPTION, TEST SET PREDICT TIME AND ACCURACY FOR DIFFERENT KNN QUANTIZATION SEGMENT NUMBER FOR LIBRIWORDS1000 DATASET.

	Memory, MB	**Time, s**	**Accuracy, %**
Basic	977	272.37	**84.2**
4	23.1	2.85	78.75
8	30.7	3.87	81
16	46	4.92	82.18
32	76.5	8.13	82.5
64	138	17.54	**82.65**

TABLE XI. MEMORY CONSUMPTION, TEST SET PREDICT TIME AND ACCURACY FOR DIFFERENT KNN QUANTIZATION SEGMENT NUMBER FOR LIBRIWORDS1000 DATASET.

	Memory, MB	**Time, s**	**Accuracy, %**
Basic	1260	440.66	**79.4**
4	29.8	4.27	68.56
8	39.6	5.69	73.33
16	59.4	7.23	75.51
32	98.8	12.03	76.35
64	178	27.15	**76.59**

TABLE XII. MEMORY CONSUMPTION, TEST SET PREDICT TIME AND ACCURACY FOR DIFFERENT KNN QUANTIZATION SEGMENT NUMBER FOR SPEECH COMMANDS V2 35 TASK.

	Memory, MB	**Time, s**	**Accuracy, %**
Basic	15.7	0.54	**96.19**
5	1.2	0.1	96.07
9	1.55	0.11	95.98

TABLE XIII. PERFORMANCE OF THE BEST KNN QUANTIZATIONS FOR DIFFERENT DATASETS

	Memory, basic	**Accuracy, basic**	**Memory, quantization**	**Accuracy, quantization**	**Memory economy, times**	**Accuracy degradation, %**
Commands	15.7	96.19	1.55	95.98	10.13	5.51
LW10	246	91.74	19.4	91.61	12.68	1.57
LW100	638	86.9	50	86.24	12.76	5.04
LW1000	977	84.2	138	82.65	7.08	9.81
LW10000	1260	79.4	178	76.59	7.08	13.64