Examining speaker and keyword uniqueness: Partitioning keyword spotting datasets for federated learning with the largest differencing method

Paul C. Wallbott[1], Sascha Grollmisch[2], and Thomas Köllmer[2]

[1] Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS
paul.wallbott@iais.fraunhofer.de
[2] Fraunhofer Institute for Digital Media Technology IDMT
{goh,kor}@idmt.fraunhofer.de

Abstract. Federated learning is a powerful training strategy for neural networks where several independent clients train a model without the need of sharing potentially sensitive data. However, real world client-local data is usually biased: A single client might have access to only a few lighting conditions in computer visions, patient groups in a hospital or speakers in a smart device performing keyword spotting. We help researchers to better understand and estimate the expected performance impacts by introducing a new method to partition a given dataset into an arbitrary amount of clients, each with unique properties, to simulate such conditions.
We apply the largest differencing method to partition the Google Speech Command dataset into clients with non-overlapping speakers and additionally unique keywords and share the script to create the novel *GSC-FL* dataset. The results, using convolutional neural networks, show that the performance of the final model is stable up to at least 16 clients and models trained only on local data are clearly outperformed by federated learning. However, unique speakers for each client have a negative performance impact and it increases even more with unique keywords. Our script can be applied with only minor adjustments to partition any other dataset for federated learning investigations as well.

Keywords: keyword spotting · federated learning · multiway number partitioning.

1 Introduction

Keyword spotting (KWS) deals with recognizing keywords such as “yes" or “stop" in a speech audio stream. A special case is the recognition of a selected word to wake-up voice assistant systems like Amazon Alexa or Microsoft Cortana (wake-word detection) before a computationally expensive automatic speech recognition is triggered to analyze the semantic meanings of longer phrases. Such wake

word or hot word detection systems can be extended by looking for several keywords which allow the hands-free control of industrial machines or the indexing of large audio archives. KWS has seen strong performance improvements with the inclusion of deep learning techniques [2, 15, 12, 11], see [10] for a recent review.

These deep learning approaches require large databases to learn robust models which are not overfitting to certain speakers or recording conditions. A simple but expensive way to improve their robustness is to collect and annotate more and more data from a vast user basis which covers different speakers and all keywords. These models can then be adapted to new keywords with few examples using transfer learning [11]. However, the centralized data collection raises several privacy, security, and logistical issues: By sharing speech data (best case with realistic background noise) also other confidential information may be shared, the centralized host needs to be fully trusted, and raw audio data (or at least a compressed input representation) needs to be transmitted. An alternative solution for obtaining robust models is Federated Learning (FL) [1]. Instead of training the model on a centralized data collection, models are trained directly on many edge devices using locally stored data. Each of these devices, so-called clients, share the parameter changes with a coordination server, which aggregates these changes to update the global model. This newer model is then transferred back to the clients and used for the next training iteration. This process is repeated until convergence or when new data is acquired [6].

The performance of federated learning systems degrades, if the data is not ideally distributed over all clients and unbalanced in terms of locally available classes. The problem of sound event detection makes no exception [5], the same goes for the KWS use case: Not all clients have the same speakers which might lead to a local speaker overfitting. Conversely, not all clients have examples for all keywords since users likely only record keywords of their interest. To the best of our knowledge, the application of FL to KWS has focused on wake-up word detection on small [9] and big [4] datasets but not considered larger keyword vocabulary nor the mentioned distribution problems.

With this paper, we propose the novel *GSC-FL* dataset which covers different realistic data distributions and publish the scripts to create it.[3] It contains pre-defined splits of the Google-Speech Commands (GSC) dataset [13] for fully random, unique speakers, and also unique keywords distributions for each client. These splits were created with the largest differencing method (LDM), see Section 2; an approach that can also be applied to other FL distribution scenarios that are unrelated to KWS, such as medical image analysis with unique patients or other sociodemographic characteristics that might bias the local models. With the *GSC-FL* dataset we answer the following research questions and provide deeper insights on FL for KWS: How is the performance of models trained with FL affected when more clients participate that have less data per client? How do unique speakers for each client affect the performance? And lastly: How do unique keywords for each client affect the performance?

[3] The scripts can be found here: `https://github.com/paul-cw/gsc-fl.git`

2 Realistic Data Distribution

To investigate the research questions, we need an appropriate FL dataset where each sample from the original dataset is assigned to a specific client. The simplest approach for GSC is to randomly select utterances of each keyword independent of the speaker from the whole set. However, for a realistic scenario, it is very unlikely that each speaker records keywords at basically every client which are normally geographically dispersed. A random distribution implicitly assumes this and following experiments might output too optimistic results. We require a more realistic data distribution that reflects particular constraints: Non-overlapping speakers between the resulting clients and the same number of recordings per client. While the former restricts the local variability in terms of speakers, we choose the latter to eliminate effects due to different number of total utterances per client. This problems arises since each speaker contributes a varying amount of keywords to the whole dataset, starting from a single word.

We can formulate the distribution problem as follows: Suppose we have N speakers in the original dataset and write down a set of integers $S = \{i_1, ..., i_N\}$ where i_n is the number of recordings that speaker n contributed to the GSC dataset. We want to split the dataset into K clients, which is then equivalent to finding K subsets of S, where the sum of elements in each subset is approximately equal. This problem is known as multiway number partitioning. While in the case of a small number of speakers one could evaluate all possible combinations directly and choose the best one, the vast amount of possible combinations makes this approach unfeasible. We propose the largest differencing method (LDM) [7] to search for an optimal solution by minimizing the difference between the subset with the largest and the one with the smallest sum. LDM consecutively replaces the two biggest numbers in S with their difference to achieve this. The resulting partitions are shown in Table 3 and discussed in the next section. To the best of our knowledge, LDM has not been considered for distributing FL datasets.

3 The GSC-FL Dataset

We start from the well known Google Speech Commands dataset, which contains 105,829 utterances of 35 keywords from 2,618 different speakers [13]. To prepare it for federated learning, we apply the following processing pipeline:

First, we select keywords: To get a balanced dataset, we select the 10 commands *yes, no, on, down, stop, right, up, go, left, off* and put the remaining keywords into the *unknown* category. For each speaker, we draw random unknown recordings until the total number of unknown keywords is equal to the speakers average number of recordings of the 10 keywords above. We create silence utterances, mix in background noise and add the same amount of unknown utterances to each speaker. The resulting dataset has 12 classes: 10 keywords, unknown, and silence.

Second, we create clients for federated learning: We assign each utterance in the training dataset[4] a client id for the splits into K=[2 ... 512] clients. This is done in two ways: For the **iid split**, we split the utterances randomly into K equal sized clients with a locally balanced amount of keywords. For the **speaker split**, we create K clients with the condition that all utterances of a given speaker go to one client only, while optimizing for identical number of recordings on each client using LDM, see Section 2. While the first split provides an optimal baseline with overlapping speakers on the different clients, the second split ensures that all utterances of a given speaker are mapped to only one client. The results in Table 3 show the roughly balanced splits with respect to the number of utterances and speakers over the resulting clients.

Third, we create unique keywords: We drop the utterances of the keywords *up, go, left, off* from all but one client (or two in the case of K=8). This corresponds to the most difficult scenario where each client has unique speakers and also partially unique keywords. The dropped utterance are also removed from the centralized dataset for a fair comparison.

The resulting datasets are shown for K=8 in Figure 1. One can see that the split is slightly more imbalanced intra-class wise for the speaker splitting but still shows a balanced data distributions. The slight increase in class imbalance is due to the non trivial task of creating K clients from the dataset. We measure the class imbalance on a client c as the deviation from the mean number of utterances per keyword:

$$\alpha_c = 100 * \frac{\sum_{k=1}^{K} |N_{ck} - N_c|}{2N_c(K-1)} \tag{1}$$

$$N_c = \frac{\sum_{k=1}^{K} N_{ck}}{K} \tag{2}$$

Where K is the number of classes on the client, N_{ck} the number of utterances of class k on client c and N_c is the average number of utterances per keyword on client c. We have normalized the measure to $\alpha_c \in [0, 100]$, where equally distributed keywords result in $\alpha_c = 0$ and unequally distributed keywords in $\alpha_c = 100$. The non iid-ness is then defined as:

$$\alpha = \frac{1}{K} \sum_{c=1}^{K} \alpha_c \tag{3}$$

where K is the number of clients and α_c defined above.

4 Experimental Setup

This section describes the input representation obtained from the raw audio data as well as the neural network architecture that is used in the following experiments.

[4] We use the predefined train, validation, test split that come with the dataset.

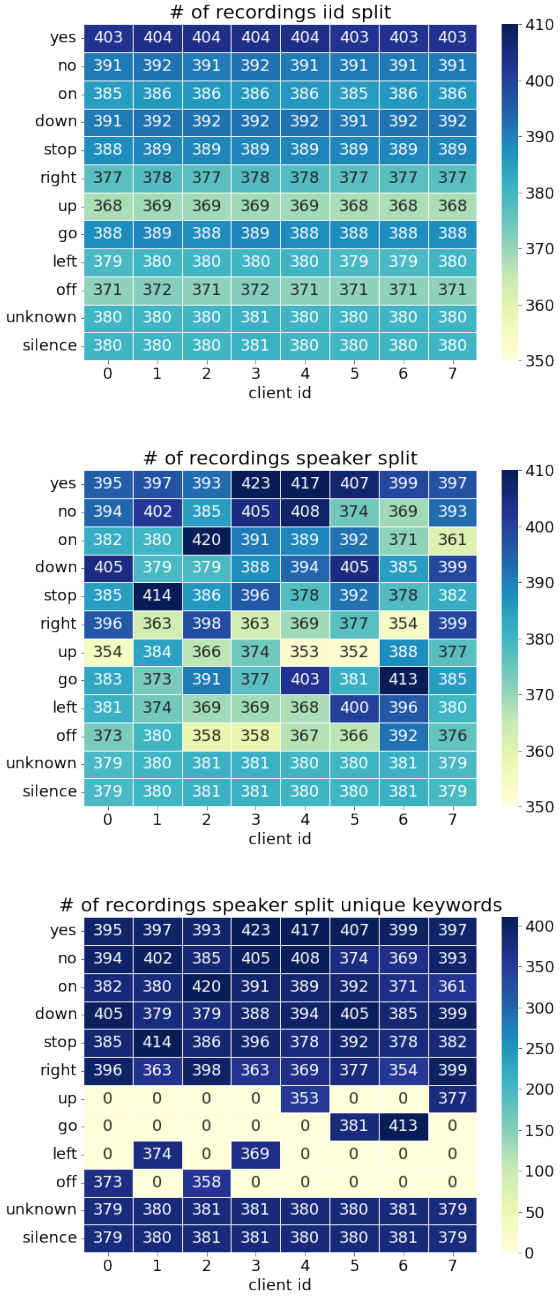

Fig. 1. Utterance distribution by keyword and client id. Color encodes the number of recordings for each combination with the different splitting methods described.

Table 1. Average dataset properties and the corresponding standard deviation after splitting into several clients in the unique speakers setup. Average speakers/utterances (utt.) are obtained by averaging the number of speakers / utterances over all clients. The non-iidness (α) is defined in equation (3). The value for the iid split is α=1.1 for all splits.

#clients (K)	speakers	utt. / kw	$\alpha_{speaker}$	α_{iid}
2	1004.0 ± 4.2	1535.5 ± 0.1	1.1 ± 0.6	1.1 ±0.0
4	502.0 ± 1.4	767.7 ± 0.0	1.4 ± 0.3	1.1 ±0.0
8	251.0 ± 1.2	383.9 ± 0.0	1.7 ± 0.3	1.1 ±0.0
16	125.5 ± 1.1	191.9 ± 0.0	2.2 ± 0.4	1.1 ±0.0
32	62.8 ± 0.8	96.0 ± 0.0	2.9 ± 0.7	1.1 ±0.0
64	31.4 ± 0.7	48.0 ± 0.0	3.8 ± 1.0	1.1 ±0.1
128	15.7±0.5	24.0 ±0.0	5.8 ±1.5	1.2 ±0.3
256	7.8 ±1.4	12.0 ±0.0	7.5 ±2.1	1.4 ±0.6
512	3.9 ±1.5	6.0 ±0.5	9.9 ±4.0	1.5 ±1.6

4.1 Input representation

The dataset contains one second long raw audio recordings with a sampling rate of 16 kHz. We extract 40 MFCC with a sliding window of $l = 40\,\text{ms}$ and a stride of $s = 20\,\text{ms}$ as in [15], resulting in a 49 x 40 dimensional input matrix for each input file. To create realistic silence utterances, we add the Google Speech Command specific background noise with a fixed amplitude to the silent utterances.

4.2 Model

For fast algorithmic iterations and the feasible execution, all experiments focus on small footprint models with less than 100k parameters. It must be noted, that the general accuracy might be increased with larger models with millions of parameters as in [11], but this is not the focus of this paper. Furthermore, these small models are more suitable for FL scenarios, where each client needs to train models locally and has likely only restricted hardware resources.

There are a variety of different small footprint models that have been tested on the Google Speech Command dataset, mostly recurrent neural network (RNN) and convolutional neural network (CNN) architectures, see [12, 15] for comparative studies of different architectures. For our purposes we use the small footprint model with temporal convolutions introduced in [3], which has only 65k parameters while achieving near state-of-the-art performance. We choose the authors' TC-ResNet8 architecture. The residual blocks contain batch normalization layers in the original formulation, which we replace with group normalization layers [14]. These were shown to be more suitable for FL [5].

4.3 Objective and training procedure

Since it is a multi-class single-label problem, we use the Categorical Cross Entropy loss to optimize the model. For the final evaluation, we pick the best training weights in terms of F-score (f1) on the fixed validation dataset. All results are reported for the pre-defined test set which is independent of the FL training data distribution. For a meaningful comparison, we hold as many parameters fixed as possible. That includes a batch size of 32 as well as the (client side) Adam optimizer [8]. We compare different learning rates and find the common 0.001 works best on average. We fix the dropout rate by running experiments with $K = 2$ and $K = 32$ clients. The former benefits from bigger dropout rates, while the latter achieves best results with no dropout. Dropout is therefore set to a rate of 0.1 as a compromise. The results could be further improved with data augmentation, hyper parameter optimization, and learning rate schedulers.

Non-FL training To understand the benefit each client can obtain from participating in the FL process, we calculate individual models on all clients in a non-federated manner. We quote the F-score on the test set of the best model from all clients in the split as the result from non-federated learning. We train for 250 epochs and use early stopping with a patience of 10 epochs. The larger the number of clients, the less data is available for individual models and we expect more gain by participating in the FL process.

FL training For federated learning we use the standard federated averaging algorithm [1] with a learning rate of 1. We run the code three times to account for randomness during training. Throughout the experiments, we train for 250 epochs (600 for the unique keyword settings due to slower convergence) and quote the F-score on the test set. We run one epoch per client for each federated averaging round. In such a round, all clients participate in the training process.

5 Results

The following section presents the results for federated compared to non-federated training and details the outcomes for the unique speakers and keyword scenarios.

5.1 FL vs. Non-FL Training

The final results for FL on the iid split are shown in Table 2. We see that FL has a positive impact compared to the local training on each client and that the difference to the non-federated F-score is positive in every case. This is expected, since the federated setup makes use of the full dataset, not only the Kth fraction as the baseline runs do. The performance of the final model is stable up to 16 clients and decreases by 4 percentage points with 64 clients. This can be attributed to the smaller amount of local data.

5.2 Unique speakers

Comparing iid and speaker splits in Table 2, one can see that the federated performance in the speaker split is roughly equal to the iid baseline for up to 16 clients and only moderately diverges up to 128 clients with 16 speakers each. With less than 10 speakers per client as in 256 and 512, the gap is increasing. One reason might be the models overfitting to local speakers since the amount of speakers is reduced with an increased number of clients. Another reason could be due to the increasing class imbalance on each client, which is shown in Table 3. This is due to the constraint of equal number of recordings and unique speakers on each client, which is harder to fulfill the more clients are used. In general, speaker overfitting might be a challenge for real world applications where one client equals one household with less than five speakers usually.

Table 2. Mean F-scores and standard deviation on the test set for iid (F_{iid}) and speaker (F_{speaker}) splits, as well as the best result for models trained locally (F_{local}) on each client without federated learning on the iid split. The best result for a fixed number of clients is bold.

#clients (K)	F_{iid} [%]	F_{speaker} [%]	F_{local} [%]
1	94.6	94.5	94.6
2	94.6 ± 0.3	**94.7 ± 0.2**	91.9
4	94.6 ± 0.3	**94.8 ± 0.6**	90.1
8	94.5 ± 0.2	**94.6 ± 0.3**	87.4
16	**94.5 ± 0.1**	94.1 ± 0.2	83.6
32	**92.9 ± 0.3**	92.5 ± 0.1	79.2
64	**91.0 ± 0.3**	90.1 ± 0.3	75.2
128	**87,7 ± 0.2**	87.0 ± 0.8	65.3
256	**84,1 ± 0.2**	80.9 ± 1.0	55.7
512	**77,7 ± 1.6**	74.8 ± 1.9	42.9

5.3 Unique keywords

Table 3. The F-scores (F) with federated learning (FL) for the splits into 4 and 8 clients with unique keywords (kws) are shown in comparison to the centralized F-scores (centr.) that are obtained using the full dataset with adjusted keyword utterances.

speaker split	centr. 4	FL 4	centr. 8	FL 8
F all kws [%]	93.1	84.5	92.6	86.8
F shared kws [%]	93.6	88.6	93.4	90.2
F unique kws [%]	92.1	76.2	91.2	80.0

The results for the unique vocabulary setting are shown in Table 3. The centralized F-scores are higher than the results achieved in the federated setup. The total amount of utterances for each keyword can be ruled out as a reason since the total utterances were also adjusted in the centralized training. This explains also the lower performance compared to Table 2. The results indicate that the missbalanced client data has a negative impact on the performance of the FL model. For 8 clients the accuracy improves since two clients have examples of each unique keyword leading to a higher training impact of the unique keywords in the averaging step of FL. As expected, the F-score is lower for the unique keywords than the shared ones that are present at each client. They also converge slower, as can be inferred from the learning curves in Figure 2. The shared vocabulary is converging after 50 epochs, while the unique keywords are reaching an F-score of 0.5. Even after the 600 epochs, federated models don't show full convergence for the unique keywords. The FL averaging process might need to be adjusted when classes are unique for some clients to address this, in future work.

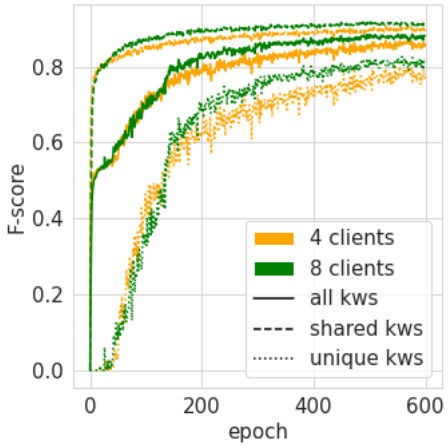

Fig. 2. The F-score on the validation set is shown for the unique keyword setting for the speaker split.

6 Conclusion

Federated Learning is a promising approach in keyword spotting use cases, where a central collection of training data is not feasible or has undesired consequences, such as possible privacy violations in case of a data breach. To facilitate research on this topic, we publish the *GSC-FL* dataset, a partitioned version of the Google Speech Command dataset into unique speakers and unique keyword splits. We

interpret the creation of such a dataset as a multiway number partitioning problem and create our dataset with the help of the largest differencing method. We suggest that this approach can be used for similar federated learning studies, where a centrally collected dataset, that contains multiple instances from the same source or any other property of interest, needs to be distributed efficiently without overlapping instances to a variable number of FL clients.

In our experiments on *GSC-FL*, we showed that the model benefits from the federated learning compared to local training, as expected. In the iid split, where speakers are present on several clients, the performance decreases with more than 16 clients and drops considerably with 512. This is caused by fewer training examples per client. Unique speakers do not influence the results negatively when more than 10 speakers contribute to each client. With only a handful of speakers, the performance degrades due to speaker overfitting and an increased class imbalance caused by the unique speaker setting. Additionally, unique keywords and therefore unique classes for each client affect the performance negatively which leaves room for improvements.

The proposed dataset and our results highlight directions for future research on federated learning in the context of KWS, such as testing different aggregation schemes, the protection against model inversion attacks, and methods to address local bias that occure with realisitically partioned datasets..

References

1. Brendan McMahan, H., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (2017)
2. Chen, G., Parada, C., Heigold, G.: Small-footprint keyword spotting using deep neural networks. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 4087–4091. IEEE, Florence, Italy (May 2014). https://doi.org/10.1109/ICASSP.2014.6854370, http://ieeexplore.ieee.org/document/6854370/
3. Choi, S., Seo, S., Shin, B., Byun, H., Kersner, M., Kim, B., Kim, D., Ha, S.: Temporal convolution for real-time keyword spotting on mobile devices. In: Kubin, G., Kacic, Z. (eds.) Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019. pp. 3372–3376. ISCA (2019). https://doi.org/10.21437/Interspeech.2019-1363, https://doi.org/10.21437/Interspeech.2019-1363
4. Hard, A., Partridge, K., Nguyen, C., Subrahmanya, N., Shah, A., Zhu, P., Lopez-Moreno, I., Mathews, R.: Training keyword spotting models on non-iid data with federated learning. In: Meng, H., Xu, B., Zheng, T.F. (eds.) Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020. pp. 4343–4347. ISCA (2020). https://doi.org/10.21437/Interspeech.2020-3023, https://doi.org/10.21437/Interspeech.2020-3023
5. Johnson, D.S., Lorenz, W., Taenzer, M., Mimilakis, S.I., Grollmisch, S., Abeßer, J., Lukashevich, H.M.: DESED-FL and URBAN-FL: federated learning datasets for sound event detection. In: 29th European Signal Processing Conference, EUSIPCO 2021, Dublin, Ireland, August 23-27, 2021. pp. 556–560. IEEE (2021). https://doi.org/10.23919/EUSIPCO54536.2021.9616102, https://doi.org/10.23919/EUSIPCO54536.2021.9616102
6. Kairouz, P., Brendan McMahan, H., et al.: Advances and Open Problems in Federated Learning. Foundations and Trends® in Machine Learning **14**(1–2), 1–210 (2021). https://doi.org/10.1561/2200000083, http://dx.doi.org/10.1561/2200000083
7. Karmarkar, N., Karp, R.M.: The differencing method of set partitioning. Computer Science Division (EECS), University of California Berkeley (1982)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980
9. Leroy, D., Coucke, A., Lavril, T., Gisselbrecht, T., Dureau, J.: Federated learning for keyword spotting. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019. pp. 6341–6345. IEEE (2019). https://doi.org/10.1109/ICASSP.2019.8683546, https://doi.org/10.1109/ICASSP.2019.8683546
10. López-Espejo, I., Tan, Z., Hansen, J.H.L., Jensen, J.: Deep spoken keyword spotting: An overview. IEEE Access **10**, 4169–4199 (2022). https://doi.org/10.1109/ACCESS.2021.3139508, https://doi.org/10.1109/ACCESS.2021.3139508

11. Mazumder, M., Banbury, C.R., Meyer, J., Warden, P., Reddi, V.J.: Few-shot keyword spotting in any language. In: Hermansky, H., Cernocký, H., Burget, L., Lamel, L., Scharenborg, O., Motlícek, P. (eds.) Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021. pp. 4214–4218. ISCA (2021). https://doi.org/10.21437/Interspeech.2021-1966, https://doi.org/10.21437/Interspeech.2021-1966
12. Rybakov, O., Kononenko, N., Subrahmanya, N., Visontai, M., Laurenzo, S.: Streaming keyword spotting on mobile devices. In: Meng, H., Xu, B., Zheng, T.F. (eds.) Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020. pp. 2277–2281. ISCA (2020). https://doi.org/10.21437/Interspeech.2020-1003, https://doi.org/10.21437/Interspeech.2020-1003
13. Warden, P.: Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. arXiv:1804.03209 [cs] (Apr 2018), http://arxiv.org/abs/1804.03209, arXiv: 1804.03209
14. Wu, Y., He, K.: Group normalization. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII. Lecture Notes in Computer Science, vol. 11217, pp. 3–19. Springer (2018). https://doi.org/10.1007/978-3-030-01261-8_1, https://doi.org/10.1007/978-3-030-01261-8_1
15. Zhang, Y., Suda, N., Lai, L., Chandra, V.: Hello Edge: Keyword Spotting on Microcontrollers. arXiv:1711.07128 [cs, eess] (Feb 2018), http://arxiv.org/abs/1711.07128, arXiv: 1711.07128