Interspeech 2022
18-22 September 2022, Incheon, Korea

Production federated keyword spotting via distillation, filtering, and joint federated-centralized training

Andrew Hard[1], *Kurt Partridge*[1], *Neng Chen*[1], *Sean Augenstein*[1], *Aishanee Shah*[1], *Hyun Jin Park*[1], *Alex Park*[1], *Sara Ng*[1,2], *Jessica Nguyen*[1], *Ignacio Lopez Moreno*[1], *Rajiv Mathews*[1], *Françoise Beaufays*[1]

[1]Google LLC, Mountain View, CA, U.S.A.
[2]University of Washington, Seattle, WA, U.S.A.

{harda,kep,nengchen,saugenst,aishaneeshah,hjpark,
axpk,sbng,jessnguyen,elnota,mathews,fsb}@google.com

Abstract

We trained a keyword spotting model using federated learning on real user devices and observed significant improvements when the model was deployed for inference on phones. To compensate for data domains that are missing from on-device training caches, we employed joint federated-centralized training. And to learn in the absence of curated labels on-device, we formulated a confidence filtering strategy based on user-feedback signals for federated distillation. These techniques created models that significantly improved quality metrics in offline evaluations and user-experience metrics in live A/B experiments.

Index Terms: federated learning, keyword spotting, filtering, distillation, multi-task learning, domain adaptation

1. Introduction

Keyword spotting (KWS) models provide an important access point for virtual assistants. *Alexa*, *Hey Google*, and *Hey Siri* are just a few examples of keywords that can be used to initiate queries and issue commands to various phone and smart speaker devices. The underlying algorithms must efficiently process streaming audio to trigger on instances of the keyword while screening out the vast majority of irrelevant noise.

Neural networks have been thoroughly studied in the context of KWS. Prior works have shown significant quality improvements and latency reductions for low-resource inference settings [1, 2, 3, 4, 5, 6].

Federated learning (FL) [7] is a privacy-enhancing distributed computation technology that can be used to improve neural models on-device without sending users' raw data to servers. Model updates are communicated from clients to the server, but they are ephemeral (only stored in temporary memory), focused (only relevant to the training task at hand), and are only released in aggregate.

Recently, FL has been applied to many production domains including keyboard next-word prediction [8] and speaker verification [9]. Prior works have also used FL algorithms to train KWS models in simulation environments [10, 11]. In most cases, federated training benefits models as a result of the abundant in-domain data available on-device.

Even so, there is often a mismatch between the domain of the cached data for FL and the inference data domain: certain device types or conditions may be underrepresented. In such cases, it is beneficial to combine FL with server-based training. One approach, federated transfer learning [12], suffers from catastrophic forgetting [13, 14]. Another more recent approach is joint training, in which server-based learning is combined with FL [15].

While supervised learning has been the dominant approach to KWS model training, semi-supervised [16, 17, 18] and self-supervised [19, 20] training techniques have become popular as they are able to leverage large unlabeled datasets to achieve comparable or superior performance to supervised models. These techniques are particularly helpful in the cases of FL (where the client cache data are "no-peek" by design and cannot be manually labeled) and speech (where audio is abundant but the labeling process is arduous). Distillation [21] of a teacher model into a student with equivalent architecture has also been used previously for KWS [22].

Filtering noisy labels has a long history and can be seen as a form of curriculum learning [23]. More recently it has been explored in semi-supervised settings in ASR [24]. See [25, 26] for more detailed surveys. However, this literature does not address how to infer label confidence using additional correlated features that are only available at training time.

This paper presents a successful application of federated learning for the speech keyword spotting domain. Several unique contributions are provided, including joint federated-centralized training and on-device example filtering based on feedback feature analysis.

2. Keyword spotting model

2.1. Input features

The input features used in this work were identical to those used in prior publications [5, 27]. A 40-dimensional vector of spectral filter-bank energies was extracted at each time frame t (generated every 10ms over a 25ms window). 3 temporally-adjacent frames were stacked and strided to produce a 120-dimensional input feature vector, X_t, every 20ms.

To improve model robustness and generalization, data augmentation techniques were applied to the features. During server training, classic augmentation methods such adding simulated reverberation and mixing noise were applied to the data prior to feature extraction, as in [28]. These methods were infeasible for on-device training [11], so spectral augmentation [29] was used instead with FL.

2.2. Architecture

The two-stage model architecture (Fig. 1) from [5, 27] was used for both the teacher and student networks. The model consisted of 7 factored convolution layers (called SVDF [27]) and 3 bot-

Copyright © 2022 ISCA

10.21437/Interspeech.2022-11050

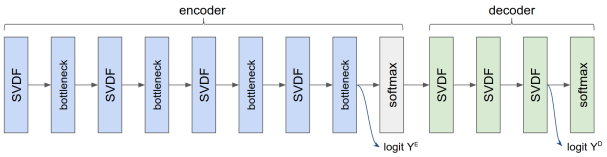

Figure 1: *Stacked SVDF with bottleneck model architecture*

tleneck projection layers, organized into sequentially-connected encoder and decoder sub-modules. In total, the model consisted of approximately 320,000 parameters.

The encoder module processed input features X_t, a vector of stacked spectral filter-bank energies. It generated an N-dimensional encoder output Y^{E} that was trained to encode N phoneme-like sound units relevant for the keyword task. The decoder module processed the encoder output and generated a 2-dimensional output Y^{D} that was trained to predict the existence of a keyword in the input audio stream. The combined prediction logit was defined as $Y = [Y^{\mathrm{E}}, Y^{\mathrm{D}}]$.

2.3. Supervised training objective

The teacher model for FL was trained in a supervised manner. Previous works proposed two types of supervised losses for KWS model training. The first loss term directly computed cross-entropy between model logits and labels [27]. The second loss term computed cross-entropy between max-pooled logits and labels [5]. Both loss terms had separate component terms for the encoder and decoder, and weighted combinations of all terms were used as the final loss.

In this work, a weighted combination of the cross-entropy and max-pooled losses was used, as the combination conferred additional regularization and generalization benefits.

$$\mathcal{L}_{\text{sup}} = L_{\text{CE}}\left(Y(X_t,\theta), c_t\right) + \alpha \cdot L_{\text{MP}}\left(Y(X_t,\theta), \omega_{\text{end}}\right) \quad (1)$$

$Y(X_t,\theta)$ represents the combined encoder and decoder model output given input X_t and parameter set θ. L_{CE} represents the end-to-end loss proposed in [27]. The implementation as defined by eq. (2) in [5] was used, where c_t is the per-frame target label for CE-loss. L_{MP} represents the max-pool loss proposed in [5], which was defined by eq. (12) in [5]. ω_{end} represents the end-of-keyword position label for the max-pool loss. α is a loss-weighting hyper-parameter determined empirically. Refer to [5, 27] for details of L_{MP} and L_{CE}.

2.4. Semi-supervised training objective

Supervised training was not possible for on-device training, since the "no-peek" nature of the data precluded manual labeling. The distillation strategy from [22] was used instead for the FL objective. A teacher model with the architecture from Section 2.2 was trained on server-based data using the supervised training objective from Section 2.3. The fully-trained teacher model, with parameter set θ^{Teacher}, was used to generate soft labels $Y^{\text{Teacher}}(X_t, \theta^{\text{Teacher}})$ using the encoder and decoder logits for each training example X_t. A student model with the same architecture as the teacher was then trained on these soft labels using the cross-entropy loss function below.

$$\mathcal{L}_{\text{semi-sup}} = L_{\text{CE}}\left(Y(X_t,\theta), Y^{\text{Teacher}}(X_t,\theta^{\text{Teacher}})\right) \quad (2)$$

3. Federated client population

Federated training data were stored in local encrypted caches on Android devices with the Google App[1]. While the app has been installed on more than 10 billion devices as of 2022 [30], caching was only enabled if users explicitly opted-in to "save audio recordings on this device and help Google improve speech technologies for everyone".

A targeted caching policy, based on the principle of data minimization [31, 32], was implemented to ensure that snippets would only be cached if they could directly improve KWS model reliability. Each client's local cache stored short snippets of voice recordings that were captured by the client whenever the Assistant activated or nearly activated, as well as metadata including how a device was configured, how and when an interaction with the Assistant happened, and whether the activation was successful [33]. Near activation events provided examples of (true and false) rejects for training, while activation events provided examples of (true and false) accepts. Clients were allowed to cache up to 20 near-activation examples per day and an unlimited number of activation examples. Training examples were retained in the cache for 63 days, after which they were automatically deleted.

When these experiments were conducted in March 2022, the median client cache contained 175 examples. These examples had an average duration of 1.7 seconds, with a maximum length determined by the size of the audio buffer [1] used for keyword spotting on-device.

Since training cache data were "no peek" by design, federated KWS training relied on semi-supervised learning techniques. Labels were inferred from a teacher model as described in Section 2.4. Additionally, on-device signals were used to filter data based on confidence in the labels (see Section 4).

Clients were only eligible for federated training if they passed certain criteria. Devices were required to have at least 2GB of memory, the Google App must have been updated within the past year, and the device must have utilized the US English language locale. Once these criteria were met, devices were only eligible for training during periods of time when they were charging, unused, connected to un-metered networks, and had non-empty training caches. For each federated computation task, there was an additional restriction that prevented any single client from contributing more than once per 24 hours.

At the time of these experiments, nearly 300,000 unique client devices checked in for training per day and completed up to 6,000 rounds of federated training per day for cohorts of 400 clients per round. Experiments in this paper took approximately 1 week to fully converge starting from random weight initialization.

4. Filtering

4.1. User feedback features

To improve the quality of the unlabeled distillation data, examples were filtered out if the teacher soft labels were likely to be incorrect. Label uncertainty was estimated using additional "feedback features". These features, which were cached with the audio, indicated whether a later stage keyword model on the server rejected the utterance, and whether the user followed up a few seconds after the utterance with a potential re-attempt of the keyword. While these feedback features indicated whether

[1] https://play.google.com/store/apps/details?id=com.google.android.googlequicksearchbox

a label might be incorrect, the features themselves were noisy.

4.2. Training cache annotation study

A user study was conducted to measure label and feedback feature accuracy for training cache examples. Users downloaded an app that played back their own cached audio snippets and asked them to indicate whether the examples contained a keyword. In total, 117 participants annotated 11,908 examples.

Heuristics were then identified that selected for utterances with high-accuracy teacher labels. Features considered for the selection heuristics included whether the example was accepted by the on-device KWS model running inference at the time of collection, the score of that KWS inference model, whether an unrecognized example had a re-attempt, whether the speaker identification model recognized the example, and whether a more powerful final-pass KWS server model accepted the example.

Table 1: *Label accuracy as inferred by on-device signals.*

Condition	Label	Accuracy
Inference KWS model rejected AND No reattempt	Negative	0.91 ± 0.01
Speaker ID model rejected AND No reattempt AND Inference Score >0.96	Positive	0.89 ± 0.04
Speaker ID model rejected AND Reattempt	Positive	0.88 ± 0.05
Server Accepted	Positive	0.99 ± 0.005

An analysis of the study data identified a set of conditions that provided a label accuracy above 88%, as shown in Table 1.

4.3. Example filtering

Cached examples that fulfilled the high-accuracy selection conditions in Table 1 were retained for training, and the remaining examples were filtered out. While this resulted in a 60% reduction in the amount of training data, the excluded data were disproportionately likely to have incorrect teacher pseudo labels.

An alternative usage of label accuracy information would be to adjust teacher model logits instead of filtering examples. However, the approach is difficult to implement in a training recipe that uses continuous soft label targets. Target alignment is challenging, given that feedback signals provide per-utterance label corrections, whereas training labels are computed on a per-frame basis.

Experiments were conducted to determine whether one-sided label adjustment was possible. Utterances were selected in which the teacher identified as containing a keyword but the label accuracy heuristics indicated otherwise. The teacher model scores for those utterances were set to zero to indicate the absence of a keyword. Unfortunately, these experiments did not yield any improvements over the filtering approach.

5. Optimization

Three general training settings were studied: pure FL, pure centralized training, and joint federated-centralized training.

5.1. Federated training

The basic federated distillation procedure from [11] was reused in this work. The distillation objective described in Section 2.4 was used for FL, in conjunction with the data filtering strategy outlined in Section 4.

Federated optimization used `FedYogi` [34], a variant of the `FedAvg` [7] algorithm which uses the adaptive Yogi optimizer [35] in lieu of averaging for the server optimizer step. At the beginning of each federated round, a cohort of 400 client devices was formed from the eligible population of devices. A set of global KWS model weights and a training plan were sent to the client devices. The clients processed local training examples in parallel, taking up to 640 steps of SGD or completing 40 local training epochs before uploading updated model weights to the server. Weight updates from all the clients in the cohort were aggregated to produce a new weight delta. The weight delta was then treated as a pseudo-gradient and used to compute a step of adaptive optimization with Yogi. This process was then repeated for up to 2,000 rounds until the model converged. Across all training rounds and clients, this procedure amounted to 512M steps of training.

Hyper-parameters were tuned using the simulation procedure described in Ref. [11]. These included the client training stopping criteria (40 local epochs or 640 steps), the optimizer configuration (Yogi with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 0.001$), clipping for client weight updates ($||L||_2 < 0.5$), softmax temperature ($T = 0.9$), client learning rate schedule (linear warm-up for 40 rounds to a maximum of 0.2, followed by exponential decay by a factor of 0.9 every 1,000 rounds), and server learning rate schedule (linear warm-up for 40 rounds to a maximum of 0.2, followed by exponential decay by a factor of 0.1 every 3,000 rounds).

5.2. Centralized training

The centralized training procedure mostly followed that described in [5], using the multi-loss training objective from Equation 1. The underlying server-hosted training data consisted of vendor-collected utterances, audio collected from YouTube videos, and utterances from logs.

Asynchronous SGD with 200 parallel trainers and 32 parameter servers was used for optimization. The learning rate was scheduled with 25M steps of linear warm-up to a maximum value of 1×10^{-5}, followed by exponential decay with a half-life of 200M steps. In total, the central training consisted of 500M steps.

5.3. Joint federated-centralized training

FL typically has advantages over centralized learning on a server, in that the training examples gathered *in situ* by edge devices are reflective of actual inference serving requests. In the case of KWS, cached examples contain real background noise, variations in speaker voices, and are generally reflective of the conditions under which users expect KWS models to function.

However, the cached data are still not fully representative of the inference data distribution, due to constraints or imbalances of *in situ* data collection. In this specific case of KWS, the data collection policy outlined in Section 3 explicitly prevented the collection of clearly negative examples. Smart speakers represented another relevant but missing data domain for KWS: although FL was conducted on phones in these experiments, the models were expected to perform inference on both phone and speaker devices.

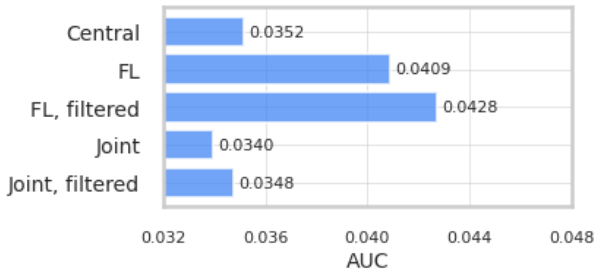

Figure 2: *AUC comparisons for different model training techniques. Lower values are better.*

The benefits of FL can be attained while overcoming any remaining train versus inference distribution skew by mixing additional central (server-based) data into a FL training process. This affords a "composite" set of training data that better matches the inference distribution. Several strategies exist for performing joint learning [15].

This work leveraged the "parallel training" strategy. In this algorithm, 500,000 steps of central training were performed in parallel with one round of FL after starting from the same global model. The model weights from these two tasks were then combined using a weighted average (central weight = 1.0, FL weight = 0.1) to form a new global model. Starting from a random model initialization, this sequence was repeated hundreds of times until the model converged. The parallel training algorithm parameters, including the averaging weights and the ratio of central steps to FL rounds, were tuned using simulations. The central and federated task components followed the descriptions in Section 5.2 and Section 5.1, respectively.

6. Experiments

Inference quality of the KWS models was assessed using offline evaluation sets and experiments on production Android phones. Comparisons were made for centrally-trained, federated, and jointly trained models, as well as models trained using the example filtering technique from Section 4.3.

6.1. Offline evaluations

Offline evaluation data were used to quantitatively measure model quality. Numerous data domains were covered, including different device types (phones and speakers), noise conditions (in-car recordings, background music), and language locales (US English, British English, etc.). The data also provided a mix of positive and negative utterances.

The number of false acceptances per hour of audio (FA/h) was computed for each model using the negative utterances, while the false reject rate per keyword instance (FRR) was measured using the positive utterances. An area under the curve (AUC) metric was defined by integrating the FR curve over the FA/h range of $[0.05, 0.5]$. Lower values were better for this threshold-independent metric.

Figure 2 compares AUC measurements for the models. Minimum values from ensembles of 5 training tasks are shown. Jointly-trained models yielded the lowest AUC, while FL-only training and example filtering led to worse AUC.

6.2. Online A/B experiments

To assess the actual user-experience of the KWS models, inference experiments were conducted on a subset of production phones running the Google App. As with the federated training caches, ground-truth labels were not available to compute metrics. Instead, proxy false accept (P_{FA}) and proxy true accept (P_{TA}) metrics were defined based on certain types of Assistant interactions following probable keywords. While both metrics were correlated with the ground-truth false accept and true accept rates, the absolute values of the metrics were not comparable. Therefore, only relative changes in the proxy metrics (ΔP_{FA} and ΔP_{TA}) were compared.

Results of production experiments are shown in Table 2. The federated and jointly-trained models significantly improved ΔP_{FA} over the control as well as the centrally-trained arm, while minimizing changes to ΔP_{TA}.

Table 2: *Relative changes in live A/B experiment metrics with respect to a centrally-trained control model. Results are presented using the 95% confidence interval for participating devices. Lower ΔP_{FA} is better, while higher ΔP_{TA} is better.*

Training task	ΔP_{FA} [%]	ΔP_{TA} [%]
Central	$[-0.91, +11]$	$[+1.5, +23]$
FL, averaged	$[-9.9, -7.4]$	$[-7.4, -0.43]$
FL, filtered	$[-11, -8.6]$	$[-5.8, +1.0]$
Joint, averaged, filtered	$[-23, -19]$	$[-10, +4.1]$

Some details should be noted. The "central" experiment arm was stopped early. This resulted in larger statistical uncertainty, and could have introduced additional systematic uncertainties not covered by the quoted 95% CI. The "Joint, averaged, filtered" experiment was conducted a month later than the others, so the confidence intervals were adjusted to account for changes to the control model. Finally, the "averaged" models were post-processed using checkpoint averaging [36].

The differences between offline and live experiments were significant. Offline AUC was likely biased in favor of centrally-trained models due to domain overlap between the training and testing data partitions. FL and filtering both under-performed on AUC but improved ΔP_{FA} and ΔP_{TA}.

7. Conclusions

We trained a keyword spotting model from scratch using federated learning and reduced mis-triggers for real users when compared with models that were centrally-trained. Improvements to the canonical FL paradigm were introduced: example filtering based on user-feedback signals improved label quality for distillation, and jointly training on federated and central data compensated for data domains that were absent in the on-device training caches. To our knowledge, this was the first user-facing launch of an FL-trained model for speech inference, the first FL model to employ contextual user-feedback signals to correct semi-supervised training labels, and the first production utilization of joint federated-centralized training.

8. Acknowledgements

The authors would like to acknowledge the contributions of Chen Chen, Katie Knister, Tamar Lucassen, Uli Sachs, Mohammadinamul Sheik, Eric Tung, Lingfeng Xu, Dan Zivkovic, and colleagues on the Google Research and Speech teams.

9. References

[1] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *ICASSP*, 2014.

[2] S. Panchapagesan, M. Sun, A. Khare, S. Matsoukas, A. Mandal, B. Hoffmeister, and S. Vitaladevuni, "Multi-task learning and weighted cross-entropy for DNN-based keyword spotting," in *INTERSPEECH*, 2016.

[3] Siri Team, "Hey siri: An on-device dnn-powered voice trigger for apple's personal assistant," https://machinelearning.apple.com/2017/10/01/hey-siri.html, accessed: 2020-04-30.

[4] M. Sun, D. Snyder, Y. Gao, V. K. Nagaraja, M. Rodehorst, S. Panchapagesan, N. Strom, S. Matsoukas, and S. Vitaladevuni, "Compressed time delay neural network for small-footprint keyword spotting," in *INTERSPEECH*, 2017.

[5] H. J. Park, P. Violette, and N. Subrahmanya, "Learning to detect keyword parts and whole by smoothed max pooling," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7899 – 7903, 2020.

[6] A. Gruenstein, R. Álvarez, C. Thornton, and M. Ghodrat, "A cascade architecture for keyword spotting on mobile devices," *ArXiv*, vol. abs/1712.03603, 2017.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[8] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *ArXiv*, vol. abs/1811.03604, 2018.

[9] F. Granqvist, M. S. Seigel, R. C. van Dalen, Á. Cahill, S. Shum, and M. Paulik, "Improving on-device speaker verification using federated learning with privacy," in *INTERSPEECH*, 2020.

[10] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[11] A. Hard, K. Partridge, C. Nguyen, N. Subrahmanya, A. Shah, P. Zhu, I. L. Moreno, and R. Mathews, "Training keyword spotting models on non-iid data with federated learning," *INTERSPEECH 2020*, pp. 4343–4347, 2020.

[12] Y. Liu, T. Chen, and Q. Yang, "Secure federated transfer learning," *ArXiv*, vol. abs/1812.03337, 2018.

[13] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[14] R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions." *Psychological review*, vol. 97 2, pp. 285–308, 1990.

[15] S. Augenstein, A. Hard, K. Partridge, and R. Mathews, "Jointly learning from decentralized (federated) and centralized data to mitigate distribution shift," *arXiv preprint arXiv:2111.12150*, 2021.

[16] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *ArXiv*, vol. abs/1807.03748, 2018.

[17] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," in *INTERSPEECH*, 2019.

[18] Y. Zhang, D. S. Park, W. Han, J. Qin, A. Gulati, J. Shor, A. Jansen, Y. Xu, Y. Huang, S. Wang, Z. Zhou, B. Li, M. Ma, W. Chan, J. Yu, Y. Wang, L. Cao, K. C. Sim, B. Ramabhadran, T. N. Sainath, F. Beaufays, Z. Chen, Q. V. Le, C.-C. Chiu, R. Pang, and Y. Wu, "Bigssl: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition," *ArXiv*, vol. abs/2109.13226, 2021.

[19] S. Löwe, P. O' Connor, and B. Veeling, "Putting an end to end-to-end: Gradient-isolated learning of representations," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[20] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 684–10 695.

[21] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *ArXiv*, vol. abs/1503.02531, 2015.

[22] H. J. Park, P. Zhu, I. L. Moreno, and N. Subrahmanya, "Noisy student-teacher training for robust keyword spotting," *INTERSPEECH 2021*, pp. 331–335, 2021.

[23] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 41–48. [Online]. Available: https://doi.org/10.1145/1553374.1553380

[24] D. Hwang, A. Misra, Z. Huo, N. Siddhartha, S. Garg, D. Qiu, K. C. Sim, T. Strohman, F. Beaufays, and Y. He, "Large-scale asr domain adaptation using self-and semi-supervised learning," *arXiv preprint arXiv:2110.00165*, 2021.

[25] B. Han, Q. Yao, T. Liu, G. Niu, I. W. Tsang, J. T. Kwok, and M. Sugiyama, "A survey of label-noise representation learning: Past, present and future," *arXiv preprint arXiv:2011.04406*, 2020.

[26] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *arXiv preprint arXiv:2007.08199*, 2020.

[27] R. Alvarez and H. J. Park, "End-to-end Streaming Keyword Spotting," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6336–6340, 2019.

[28] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. Sainath, and M. Bacchiani, "Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in google home," *INTERSPEECH 2017*, pp. 379–383, 2017.

[29] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le:, "Specaugment: A simple data augmentation method for automatic speech recognition," *INTERSPEECH 2019*, pp. 2613–2617, 2019.

[30] "Google app," https://play.google.com/store/apps/details?id=com.google.android.googlequicksearchbox, accessed: 2022-03-08.

[31] The White House, "Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy," pp. 93–135, 01 2013.

[32] "Article 5 gdpr: Principles relating to processing of personal data," https://gdpr-info.eu/art-5-gdpr/, accessed: 2022-03-08.

[33] "Your voice & audio data stays private while google assistant improves," https://support.google.com/assistant/answer/10176224, accessed: 2022-03-14.

[34] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive Federated Optimization," *arXiv e-prints*, Feb. 2020.

[35] M. Zaheer, S. J. Reddi, D. Sachan, S. Kale, and S. Kumar, "Adaptive methods for nonconvex optimization," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018.

[36] J. Utans, "Weight averaging for neural networks and local resampling schemes," in *AAAI-96 Workshop on Integrating Multiple Learned Models*. AAAI Press, 1996, pp. 133–138.