Interspeech 2022
18-22 September 2022, Incheon, Korea

FedNST: Federated Noisy Student Training for Automatic Speech Recognition

Haaris Mehmood, Agnieszka Dobrowolska, Karthikeyan Saravanan, Mete Ozay

Samsung Research UK

Abstract

Federated Learning (FL) enables training state-of-the-art Automatic Speech Recognition (ASR) models on user devices (clients) in distributed systems, hence preventing transmission of raw user data to a central server. A key challenge facing practical adoption of FL for ASR is obtaining ground-truth labels on the clients. Existing approaches rely on clients to manually transcribe their speech, which is impractical for obtaining large training corpora. A promising alternative is using semi-/self-supervised learning approaches to leverage unlabelled user data. To this end, we propose FEDNST, a novel method for training distributed ASR models using private and unlabelled user data. We explore various facets of FEDNST, such as training models with different proportions of labelled and unlabelled data, and evaluate the proposed approach on 1173 simulated clients. Evaluating FEDNST on LibriSpeech, where 960 hours of speech data is split equally into server (labelled) and client (unlabelled) data, showed a **22.5% relative word error rate reduction** (WERR) over a supervised baseline trained only on server data.

Index Terms: Federated Learning, Speech Recognition, Semi-supervised Learning, Self-training.

1. Introduction

Significant improvements in Automatic Speech Recognition (ASR) have been achieved through the development of End-to-End (E2E) Attention-based models [1, 2] and semi/self-supervised learning [3, 4], allowing for utilization of ever-increasing training corpora. Acquiring speech data for applications such as voice assistants requires transferring sensitive user data to the cloud, leading to privacy compromises [5, 6, 7, 8]. Federated Learning (FL) [9, 10] offers a solution by training models on user devices (clients) without sharing private data with the server. Briefly, an FL algorithm involves: (1) selecting a group of clients, (2) transmitting a *global* model to clients, (3) training the global model on local user data, (4) transmitting *gradients/weights* back to the server, (5) aggregating *gradients/weights*, and (6) repeating steps 1-5 until convergence.

Various Federated ASR methods have been proposed to train ASR models in FL systems [11, 12, 13, 14]. Specific challenges arising from data heterogeneity (speech characteristics, amount of data, acoustic environments etc.) are addressed via client-dependent data transformations [14] and imposing upper limits on the number of client samples [13]. Improvements to distributed optimization of models, such as alternative aggregation weighting schemes based on Word Error Rate (WER) [12] and hierarchical gradients [11] have also been proposed. A realistic setup for Federated ASR is presented in [12], showing feasibility with the French and Italian CommonVoice subsets [15], comprising of thousands of challenging speakers. The above approaches all assume availability of labelled data on clients participating in FL. In real-world ASR applications, however, manual annotation of user data on the clients is infeasible.

Recent semi- and self-supervised central training methods achieve state-of-the-art (SotA) accuracy, but require storing data at a central location [16, 2, 17]. A natural question then arises: can we adopt vanilla methods to leverage *unlabelled* user data for federated training of ASR models?

Semi-/self-supervised learning methods have been explored in FL systems for image and audio classification tasks [18, 19, 20, 21, 22]. However, these methods cannot be directly applied to tackle the above problem, since the proposed objectives are not applicable for sequence-to-sequence learning.

A semi-supervised FL method proposed for ASR [23] involves uploading unlabelled speech data from clients to cloud storage. An ASR model is then trained on this data using a federation of a few model trainers with high computational resources. The proposed approach is effective in a *cross-silo* setup, but it is not applicable to a *cross-device* setup – a federation of thousands of user devices with low computational resources.

To this end, we propose a new method called Federated Noisy Student Training (FEDNST), leveraging unlabelled speech data from clients to improve ASR models by adapting Noisy Student Training (NST) [24] for FL. Our work explores a challenging scenario: each client holds and trains a model exclusively on its own unlabelled speech data, leading to a heterogeneous data distribution, and, more than a thousand clients participate in FL, resulting in a cross-device scenario.

The contributions of this work are as follows:

- To our best knowledge, this is the first work which aims to leverage private unlabelled speech data distributed amongst thousands of clients to improve accuracy of end-to-end ASR models in FL systems. For this purpose, we propose a new method called FEDNST, employing noisy student training for federated ASR models, which achieves **22.5% WERR over training with only labelled data.**
- We elucidate the change in WER of ASR models from central training to cross-device Federated Learning regimes with FEDNST, achieving a **marginal 2.2% relative difference from fully-centralized NST** in a comparable setup.

2. Background

End-to-End (E2E) ASR can be viewed as translating a sequence of input audio frames $\mathbf{x} = (x_1, \ldots, x_T)$, into a sequence of corresponding labels $\mathbf{y} = (y_1, \ldots, y_L)$.

Modern E2E ASR models consist of an encoder-decoder architecture, trained using a weighted sum of the sequence-to-sequence (Seq2Seq) [25] objective $\mathcal{L}_{\text{Seq2Seq}}$ and the Connectionist Temporal Classification (CTC) [26] objective $\mathcal{L}_{\text{CTC}}$:

$$\mathcal{L} = \nu\mathcal{L}_{\text{CTC}} + (1 - \nu)\mathcal{L}_{\text{Seq2Seq}} \tag{1}$$

where $\nu \in [0, 1]$. While the CTC objective helps with convergence during the early stages of training and is more robust to

10.21437/Interspeech.2022-252

Algorithm 1 FedNST

S indicates a server variable and C indicates a client variable.

```
Input: $\theta_0$, ClientOpt$_{C,S}$, ServerOpt, $\eta_{C,S}$, $\eta$, $T$, $E$
for each client $i \in \mathcal{C}$ in parallel do
    PseudoLabel$(\theta_0)$                              ▷ generates $\mathcal{D}^i_{\hat{U}}$
for $t = 0, \dots, T-1$ do
    $\eta^t_C = \eta^{t-1}_C \mu^{t/\lambda}$                    ▷ Init. $\eta^{-1}_C = \eta_C$
    Sample a subset $\mathcal{S}_t \subseteq \mathcal{C}$ of clients
    for each client $i \in \mathcal{S}_t$ in parallel do
        $\theta^t_i = \theta_t, \eta^t_i = \eta^t_C$              ▷ receive $\theta_t, \eta^t_C$ from server
        Retrieve: $\mathcal{D}^i_{\hat{U}}$
        for $e = 0, \dots, E-1$ do
            for $b \in \mathcal{B}_i \sim \mathcal{D}^i_{\hat{U}}$ do
                $g_i = \nabla \mathcal{L}_i(\theta^t_i; b)$
                $\theta^t_i =$ ClientOpt$_C(\theta^t_i, g_i, \eta^t_i, e|\mathcal{B}_i| + b)$
        $\Delta^t_i = \theta^t_i - \theta_t, n_i = |\mathcal{D}^i_{\hat{U}}|$       ▷ send $\Delta^t_i$, $n_i$ to server
    $n = \sum_{i \in \mathcal{S}_t} n_i$
    $\Delta^t_C = \sum_{i \in \mathcal{S}} \frac{n_i}{n} \Delta^t_i$
    $\theta^t_S = \theta_t$
    for $b \in \mathcal{B}_S \sim \mathcal{D}_L$ do
        $g_S = \nabla \mathcal{L}_S(\theta^t_S; b)$
        $\theta^t_S =$ ClientOpt$_S(\theta^t_S, g_S, \eta_S, b)$
    $\Delta^t_S = \theta^t_S - \theta_t$
    $\Delta_t = \alpha \Delta^t_S + (1-\alpha)\Delta^t_C$
    $\theta_{t+1} =$ ServerOpt$(x_t, -\Delta_t, \eta, t)$
```

noisy conditions, the attention-based Seq2Seq objective helps in understanding long-range dependencies [27]. Recent advancements in E2E ASR models introduce self-attention to better capture intra-sequence relationships [1].

Noisy Student Training (NST) for ASR, a semi-supervised learning algorithm originally proposed for image classification [24], has been recently shown to significantly improve ASR performance [4]. Briefly, NST [4] involves: (1) training an initial model θ_0 on a labelled dataset $\mathcal{D}_L$; (2) integrating θ_0 with a language model; (3) generating pseudo-labels for an unlabelled dataset $\mathcal{D}_U$ with θ_0 (here data filtering and balancing may be applied); (4) training θ_0 on a mix of $\mathcal{D}_L$ and $\mathcal{D}_U$ to generate θ_1; (5) repeating steps 1-4 until convergence.

3. Proposed Approach

Federated Noisy Student Training (FedNST) considers a scenario where a corpus of labeled data $\mathcal{D}_L$ is available on a central server. Each client $i \in \mathcal{C}$, where $\mathcal{C}$ is the set of all clients, has an unlabelled speech dataset $\mathcal{D}^i_U$. The aim of FedNST is then to leverage the unlabelled datasets $\mathcal{D}^i_U, \forall i \in \mathcal{C}$, to improve accuracy of an ASR model trained on $\mathcal{D}_L$ while preserving user privacy, i.e., without sending user data to the server.

3.1. Federated Noisy Student Training

Algorithm 1 describes training models using FedNST which requires an initial ASR model trained using some labelled data. This requirement is supported by several studies [28, 29, 12] which discuss the complexity of training modern SotA architectures from scratch with FL.

Thus, following the standard training procedure described in Section 2, a baseline ASR model is first trained on a central server using $\mathcal{D}_L$ to produce θ_0. Next, the model θ_0 is transferred to all clients $\mathcal{C}$. The clients use θ_0 to *pseudo-label* their data $\mathcal{D}^i_U$ via PseudoLabel (Algorithm 2). A language model (LM) ϕ is integrated into the ASR model and a beam search [30] is used to improve the quality of the transcripts.

Algorithm 2 PseudoLabel(θ) - for every client $i \in \mathcal{C}$.

```
Input from server: $\theta$
Retrieve: $\phi$, $\mathcal{D}^i_U$
Initialize: $\mathcal{D}^i_{\hat{U}} = \varnothing$
for $j = 1, 2, \dots, |\mathcal{D}^i_U|$ do
    $\hat{\mathbf{y}}^i_j = f(\mathbf{x}^i_j; \theta; \phi)$
    $\mathcal{D}^i_{\hat{U}} \leftarrow (\mathbf{x}^i_j, \hat{\mathbf{y}}^i_j)$
Store $\mathcal{D}^i_{\hat{U}}$ on client $i$ for future retrieval
```

Once all clients C have pseudo-labelled their data, at each FL training round t, the server transfers it's latest model θ_t to a randomly sampled fraction of clients $\mathcal{S}_t \subseteq \mathcal{C}$. Next, each *client* $i \in \mathcal{S}_t$ trains on their pseudo-labelled data, generating a new model, θ_i (see Algorithm 1). The updated models are then transferred back to the server and aggregated to generate a new global model θ_{t+1}. This process is repeated for T rounds or until convergence. The optimization procedure is described in detail in Section 3.2 and summarized in Algorithm 1.

We explore design choices for FedNST which balance model performance and training cost on two fronts: when to pseudo-label and whether to re-use labelled server data.

First, instead of sending the model θ_0 to all clients $\mathcal{C}$ at the beginning to perform PseudoLabel, we propose an alternative strategy: at every round t, the participating clients $\mathcal{S}_t$ perform pseudo-labelling using the latest global model, θ_t.

Secondly, the standard NST [4] procedure described in Section 2 involves *mixing* samples from the labelled and pseudo-labelled datasets for each training step. However, in supervised FL, it is common to start with a model pre-trained on the server with some labelled data, and during the FL process to only use client data for model updates [11, 12]. In this work, we perform an empirical study to determine the optimal strategies for data mixing with FedNST– only aggregating client updates, or also incorporating the supervised data on the server, which was used for pre-training θ_0.

3.2. Federated Optimization for FedNST

The optimization procedure described in Algorithm 1, adopts the notation introduced in [31]. $\mathcal{L}(\cdot)$ denotes the ASR loss defined in (1). We define two optimizers [31], ClientOpt and ServerOpt, which are used on the clients and on the server, respectively. Each performs a single step of gradient descent based on the weights θ, gradients g (or pseudo-gradients $-\Delta$), learning rate η and optimization step-count. All clients individually train on their own pseudo-labelled data using ClientOpt$_C$ without sending this data to the server. The optimizer ClientOpt$_S$ is used to train on the labelled data which exists on the server.

For each round t in FL, the same model θ_t is updated in parallel on both the server and on a set of randomly sampled clients $\mathcal{S}_t$. The resulting *weights difference* $\Delta^t_i = \theta^t_i - \theta_t$ is uploaded to the server by each participating client i after their local training for E epochs. This is aggregated by the server to give Δ^t_C. Similarly, a *weights difference* Δ^t_S is produced after server training. The hyper-parameter α performs a weighted average of Δ^t_C and Δ^t_S to produce *pseudo-gradients* $-\Delta_t$. Next, $-\Delta_t$ is passed to ServerOpt to produce a set of weights for the next round θ_{t+1}. This process is repeated for a pre-determined number of rounds, T, or until convergence of models as measured via WER on a validation set located on the server.

4. Methodology

Evaluation Datasets: We evaluate our proposed method on the LibriSpeech (LS) dataset [32]. Following the setup described in [11], we divide the dataset into two equal subsets, $\mathcal{A}$ and $\mathcal{B}$, each comprising 480h. The two sets are *disjoint*, such that all data arising from a single speaker is assigned to only one set.

In the experimental analyses, we explore two scenarios:

1. Set $\mathcal{A}$ is used only for initial model pre-training, i.e., an FL step is performed using only set $\mathcal{B}$ (no data mixing).
2. Set $\mathcal{A}$ is mixed with $\mathcal{B}$ during federated training of models as discussed in Algorithm 1.

Experimental Setup: We use an end-to-end ASR model architecture [33] with a Conformer (S) encoder [1], a transformer decoder and a joint CTC+Seq2Seq objective. For Algorithm 1, we set $T = 1000$, $E = 1$, $\eta_{C,S} = 0.1$, $\eta = 1.0$, $\alpha = 0.5$ and $\frac{|\mathcal{S}_t|}{|\mathcal{C}|} = 7\%$. We set an equivalent number of epochs for the centralized experiments. After training, the model with the lowest WER on the dev-clean is selected. Results reported on the dev-clean, test-clean and test-other sets of LS are obtained by fusing the ASR model with an off-the-shelf language model [33] and using a beam search of size 10. For the federated experiments, we use a FL simulation platform with design characteristics similar to those described in [11].

Batch Normalization for FEDNST: As reported in other works [34, 35, 36], we find that using standard Batch Normalization (BN) for FL gives rise to convergence issues due to data heterogeneity. To address this issue, we replace all BN layers with a modified version of static Batch Normalization (sBN) [35]. sBN does not keep track of running statistics, i.e., the moving-average mean and variance associated with BN layers, during FL training. At the end of FL training, it queries all clients sequentially to produce global BN statistics which can then be used to evaluate the trained model. This post-processing step has computational and privacy concerns [35] and it makes it difficult to perform model evaluation during training.

Our modification is to simply re-use the running statistics from θ_0 – the model trained using the supervised data corpus $\mathcal{D}_L$ located at the server. The behaviour of our modified sBN layers during FL training is the same as original BN: normalization using batch mean and variance, followed by re-scale and shift using the trainable parameters γ and β. However, it is worth noting that this approach assumes that the client and server data arise from the same data distribution i.e. we expect running statistics before and after FL to be the same.

Federated Optimization in FedNST: Using the FEDOPT formulation for federated optimization [31], we examined the use of adaptive (FEDAVGM and FEDADAM) and standard (FEDAVG [10]) server-side optimizers for FL. We found that the differences in accuracy between these optimizers were limited (results not shown due to limited space), and hence choose the computationally more efficient FEDAVG in our study.

For the client-side optimizer, our default configuration [33] uses Adam to train the model. Our experiments indicate that using Adam as a local optimizer performs worse than using SGD because a cross-device FL system expects stateless clients, whereas Adam has stateful parameters. This was also found to be the case in [11]. Thus, we use the SGD optimizer for clients. We set α defined in Algorithm 1 to 0.5 for all experiments as we found this to be the best value.

Learning Rate (LR) Decay for Clients in FedNST: When data is distributed across clients (e.g. one client per speaker), such a dataset is considered to be non-iid. Li et al. [37] reported that the LR of federated optimizers must decay to guarantee convergence when training with non-iid data. Thus, we use LR decay to seek a better global minima in the analyses. Among various LR reduction methods [38], we choose to only decay the client LR η_C^t at the start of each round t and keep it fixed for local training. We use exponential decay as described in Algorithm 1 where the LR η_C^t is shared amongst participating clients $\mathcal{S}_t$, and updated at each round. We experimented with various values for μ (decay rate) and λ (decay steps) defining the LR decay curve. Training curves used for selecting a suitable value of λ are shown in Figure 1. We found $\lambda = 1000$ to perform the best, and used this for all subsequent experiments. Therefore, for our setup, using a weak decay is more effective than strong decay.

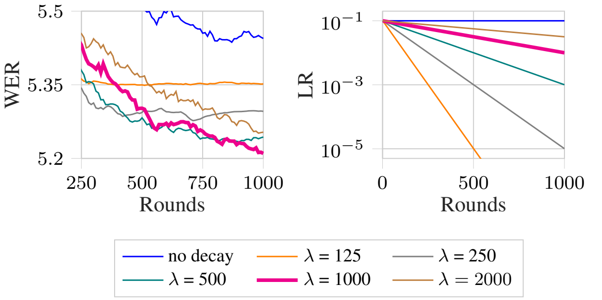

Figure 1: WER of models on the dev-clean (greedy search and no LM) (left) and client LR (right) for various λ values. Plot on the left is smoothed using exponential moving average with weight 0.8 and only the last 750 rounds are shown. Plot on the right is logarithmically scaled along the y-axis.

Table 1: *Description of data splits used for experiments.*

Split	Hrs	Num. Spks	Supervision	Location
$\mathcal{A}$	480	1165	Labelled	Server
$\mathcal{B}$	480	1173	Unlabelled	Clients

5. Experimental Analyses and Results

5.1. Comparison with SotA Methods

In Table 2, we compare WER of models trained using centralized supervised learning (SL), supervised Federated Learning (SFL), NST, and our proposed FEDNST.

The results given in the first row (indicated by SL$_{\text{seed}}$) are obtained through training models on $\mathcal{A}$ (labelled). The results given in the second row (indicated by SL) are obtained by training models on $\mathcal{A} \cup \mathcal{B}$ (both labelled). Here, SL is the lower-bound or best possible WER. We present 2.59 % WER on test-clean with 960h of labelled data which is close to SotA for a comparable sized model [1]. The last three rows present, NST, SFL and FEDNST, trained over a combination of $\mathcal{A}$ and $\mathcal{B}$, starting from the pre-trained model SL$_{\text{seed}}$. All FEDNST experiments use SL$_{seed}$ as θ_0 from Table 2 unless stated otherwise. For simplicity, in our study, both NST and FEDNST models are trained for a single generation and without any data filtering or balancing methods defined in [4].

FEDNST achieves comparable WER over NST (2.2% relative difference for test-clean), without the need of sending user data to a central server. We further observe a relative *WER increase* of 8.5% when comparing SFL with FEDNST. This is still smaller than the 22.5% relative *WER reduction* achieved by FEDNST over SL$_{seed}$. These results provide strong motivations to use FEDNST in a real-world scenario.

Table 2: *Comparison of WER (%) of models trained under different labelled data regimes.*

Method	L	U	test-clean	test-other
SL_{seed}	$\mathcal{A}$	$\varnothing$	4.27	8.99
SL	$\mathcal{A} \cup \mathcal{B}$	$\varnothing$	2.59	6.30
NST	$\mathcal{A}$	$\mathcal{B}$	**3.24**	**7.96**
SFL	$\mathcal{A} \cup \mathcal{B}$	$\varnothing$	3.05	7.57
FEDNST	$\mathcal{A}$	$\mathcal{B}$	**3.31**	**8.07**

Table 3: *An analysis of the relationship between the duration of labelled data $|\mathcal{A}|$ in hours and WER (%) of models.*

$\lvert\mathcal{A}\rvert$	SL_{seed}		FEDNST	
	test-clean	test-other	test-clean	test-other
120	9.44	18.0	7.89	15.9
240	6.96	12.9	5.30	12.0
360	5.52	11.0	4.69	10.0
480	4.27	8.99	3.31	8.07

5.2. Analyses with Different Pseudo-Labelled Data Regimes

Intuitively, models trained with fully supervised data perform better than those trained with self-training methods (e.g. NST) on partially labelled data. This is due to the noise induced by inaccurately predicted labels. We explore how changing the proportion of labelled and pseudo-labelled data affects model performance. The purpose is to: (1) quantify the hypothetical gain in performance if ground-truth labels were available for all clients, and (2) empirically verify if $\mathcal{A}$ is useful during federated training. The results are depicted in Figure 2.

Figure 2 presents two analyses: Figure 2 (i) varies the percentage of clients with entirely labelled data against clients which pseudo-label their data (i.e., applying Algorithm 2). Figure 2 (ii) varies percentage of labelled samples per client, i.e., ratio of samples with ground-truth against samples that are pseudo-labelled *for each client*. Each plot in Figure 2 presents two sets of graphs showing results from test-clean and test-other, respectively. Within each set, graphs representing $\mathcal{A} \cup \mathcal{B}$ (blue) and $\mathcal{B}$ (red) show the impact of dataset mixing in FL. Percentage of labelled clients and labelled samples per client is thereafter referred to as *'x%-labelled'* for brevity.

Figures 2 (i) and (ii) show similar trends, since in both cases the cumulative hours of labelled vs. unlabelled data obtained from all clients are roughly the same. With reference to (1), we find that *0%-labelled* leads to a relative WER increase of 8.5% compared to *100%-labelled* for the $\mathcal{A} \cup \mathcal{B}$ setup. With reference to (2), training models with FedNST using only $\mathcal{B}$ does not cause significant catastrophic forgetting. Instead, the model retains its ability to further learn from new data. We find that when using *0%-labelled*, re-using $\mathcal{A}$ (i.e., data-mixing), produces 0.5 WER improvement. However, this improvement reduces to nearly 0 when at least 25% of the data is labelled.

We hypothesize that the benefits seen with $\mathcal{A} \cup \mathcal{B}$ at *0%-labelled* come from $\mathcal{A}$ acting as a form of regularisation when most of the client data is pseudo-labelled. Exploring continual learning techniques for this problem is a promising future direction. It is worth noting the surprisingly small difference in WER seen with and without mixing labelled and unlabelled data – this is likely due to the nature of LS rather than a general observation for ASR datasets (as discussed in [11, 12]).

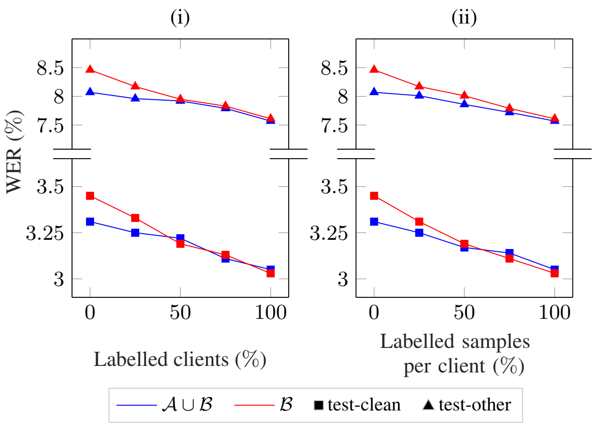

Figure 2: Reduction of WER as the percentage of labelled clients (i) or samples per client (ii) increases. Relative WER increases from labelled (100%) to pseudo-labelled (0%) data regime.

5.3. Analyses with Different Labelled Data Regimes

We explore the effect of changing the amount of labelled data $\mathcal{A}$ to produce SL_{seed} and further with pseudo-labelled $\mathcal{B}$ with FEDNST in Table 3. We used $|\mathcal{A}| \in \{120, 240, 360, 480\}$ hours and $|\mathcal{B}| = 480$ hours for all experiments. As expected, decreasing available labelled data for pre-training shows higher WER compared to FEDNST results in Table 2. Table 3 also points to a clear correlation between the WER of the initial pre-trained model and that of FEDNST.

5.4. Analyses with Varying Pseudo-Labelling Frequency

We ran an experiment to test our alternative pseudo-labelling strategy in which PSEUDOLABEL is performed every round t for clients $\mathcal{S}_t$. We provide the results in Table 4. Although this strategy performs very similarly to our original approach in terms of WER, the pseudo-labelling step is highly computationally expensive, increasing the overall FL experiment time by 10x. We leave optimizing on this front as future work.

Table 4: *Comparison of pseudo-labelling strategies, using labelled $\mathcal{A}$ and pseudo-labelled $\mathcal{B}$ for 50 rounds.*

Labelling	test-clean	test-other	Wall-clock/FL Round (Mins)
Once	3.31	8.07	1.74
Every round	3.34	8.29	18.6

6. Conclusions

We have proposed a new method called FEDNST for semi-supervised training of ASR models in FL systems. FEDNST performs noisy student training to leverage private unlabelled user data and improves the accuracy of models in low-labelled data regimes using FL. Evaluating FEDNST on real-world ASR use-cases using the LibriSpeech dataset with over 1000 simulated FL clients showed **22.5% relative WERR** over a supervised baseline trained only with labelled data available at the server. Our analyses showed that FEDNST achieves a WER comparable to fully centralized NST and to supervised training while incurring **no extra communication overhead** compared to FEDAVG. In the future, we plan to employ FEDNST on more challenging datasets, e.g., CommonVoice [15], and to incorporate other methods to learn from unlabelled data, such as Wav2Vec2.0 [17].

7. References

[1] A. Gulati, J. Qin, C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *Interspeech*, 2020, pp. 5036–5040.

[2] Y. Zhang, J. Qin, D. S. Park, W. Han, C. Chiu, R. Pang, Q. V. Le, and Y. Wu, "Pushing the limits of semi-supervised learning for automatic speech recognition," *arXiv*, vol. abs/2010.10504, 2020.

[3] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, no. 2, pp. 373–440, 2020.

[4] D. S. Park, Y. Zhang, Y. Jia, W. Han, C. Chiu, B. Li, Y. Wu, and Q. V. Le, "Improved noisy student training for automatic speech recognition," in *Interspeech*, 2020, pp. 2817–2821.

[5] B. W. Schuller, S. Steidl, A. Batliner, E. Nöth, A. Vinciarelli, F. Burkhardt, R. van Son, F. Weninger, F. Eyben, T. Bocklet, G. Mohammadi, and B. Weiss, "A survey on perceived speaker traits: Personality, likability, pathology, and the first challenge," *Computer Speech and Language*, vol. 29, no. 1, pp. 100–131, 2015.

[6] Y. Gu, X. Li, S. Chen, J. Zhang, and I. Marsic, "Speech intention classification with multimodal deep learning," in *Canadian Conference on Artificial Intelligence*, vol. 10233, 2017, pp. 260–271.

[7] M. Kotti and C. Kotropoulos, "Gender classification in two emotional speech databases," in *ICPR*, 2008, pp. 1–4.

[8] B. M. L. Srivastava, A. Bellet, M. Tommasi, and E. Vincent, "Privacy-preserving adversarial representation learning in ASR: reality or illusion?" in *Interspeech*, 2019, pp. 3700–3704.

[9] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv*, vol. abs/1610.05492, 2016.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, vol. 54, 2017, pp. 1273–1282.

[11] D. Dimitriadis, K. Kumatani, R. Gmyr, Y. Gaur, and S. E. Eskimez, "A federated approach in training acoustic models," in *Interspeech*, 2020, pp. 981–985.

[12] Y. Gao, T. Parcollet, S. Zaiem, J. Fernández-Marqués, P. P. B. de Gusmao, D. J. Beutel, and N. D. Lane, "End-to-end speech recognition from federated acoustic models," in *ICASSP*, 2022, pp. 7227–7231.

[13] D. Guliani, F. Beaufays, and G. Motta, "Training speech recognition models with federated learning: A quality/cost framework," in *ICASSP*, 2021, pp. 3080–3084.

[14] X. Cui, S. Lu, and B. Kingsbury, "Federated acoustic modeling for automatic speech recognition," in *ICASSP*, 2021, pp. 6748–6752.

[15] R. Ardila, M. Branson, K. Davis, M. Kohler, J. Meyer, M. Henretty, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," in *LREC*, 2020, pp. 4218–4222.

[16] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *NeurIPS*, 2020.

[17] Y. Chung, Y. Zhang, W. Han, C. Chiu, J. Qin, R. Pang, and Y. Wu, "w2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training," in *ASRU*, 2021, pp. 244–250.

[18] J. Kahn, A. Lee, and A. Hannun, "Self-training for end-to-end speech recognition," in *ICASSP*, 2020, pp. 7084–7088.

[19] W. Jeong, J. Yoon, E. Yang, and S. J. Hwang, "Federated semi-supervised learning with inter-client consistency & disjoint learning," in *ICLR*, 2021.

[20] H. Lin, J. Lou, L. Xiong, and C. Shahabi, "Semifed: Semi-supervised federated learning with consistency and pseudo-labeling," *arXiv*, vol. abs/2108.09412, 2021.

[21] Z. Zhang, S. Ma, J. Nie, Y. Wu, Q. Yan, X. Xu, and D. Niyato, "Semi-supervised federated learning with non-iid data: Algorithm and system design," in *HPCC/DSS/SmartCity/DependSys*, 2021, pp. 157–164.

[22] W. Zhuang, Y. Wen, and S. Zhang, "Divergence-aware federated self-supervised learning," in *ICLR*, 2022.

[23] K. Nandury, A. Mohan, and F. Weber, "Cross-silo federated training in the cloud with diversity scaling and semi-supervised learning," in *ICASSP*, 2021, pp. 3085–3089.

[24] Q. Xie, M. Luong, E. H. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *CVPR*, 2020, pp. 10 684–10 695.

[25] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *ICASSP*, 2016, pp. 4945–4949.

[26] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *ICML*, vol. 32, 2014, pp. 1764–1772.

[27] S. Kim, T. Hori, and S. Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in *ICASSP*, 2017, pp. 4835–4839.

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

[29] S. Bansal, H. Kamper, K. Livescu, A. Lopez, and S. Goldwater, "Pre-training on high-resource speech recognition improves low-resource speech-to-text translation," in *NAACL-HLT*, 2019, pp. 58–68.

[30] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014, pp. 3104–3112.

[31] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *ICLR*, 2021.

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

[33] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J. Chou, S. Yeh, S. Fu, C. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. D. Mori, and Y. Bengio, "Speechbrain: A general-purpose speech toolkit," *arXiv*, vol. abs/2106.04624, 2021.

[34] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *ICML*, vol. 119, 2020, pp. 4387–4398.

[35] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," in *ICLR*, 2021.

[36] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "Fedbn: Federated learning on non-iid features via local batch normalization," in *ICLR*, 2021.

[37] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *ICLR*, 2020.

[38] Z. Charles and J. Konečný, "On the outsized importance of learning rates in local update methods," *arXiv*, vol. abs/2007.00878, 2020.