 doi:10.20944/preprints202209.0435.v1

Article

Ensemble Weighting Strategy For Federated Learning To Handle Heterogeneous Data Distributions

Lucas Richter [1,*], Ilja Dontsov [2,*] and Tania Jacob [2,*]

[1] Fraunhofer IOSB-AST; https://www.iosb-ast.fraunhofer.de
[2] Fraunhofer ITWM; https://www.itwm.fraunhofer.de
* Correspondence: lucas.richter@iosb-ast.fraunhofer.de

Abstract: Increasingly measured data in the context of smart cities can be used to develop new and innovative business models to increase efficiency and the value of life. A time-series classification algorithm can support to automatize many different processes such as forecasting services. In order to ensure data security and privacy, Federated Learning (*FL*) trains a global model collaboratively on multiple clients. Having different data-distributions (*dd*) and data-quantities (*dq*) across participating clients, neural networks suffer from slow convergence and overfitting [2]. Based on different *dd*, *dq* and number of clients, we develop and evaluate different data-clustering strategies to update global model weights in comparison to the state of the art [9]. We use public time-series data from [1], generate various synthetic datasets considering *dd*, *dq* and train a Relational-Regularized Autoencoder (*RAE*) for classification purposes. Our results show an improvement of model performance concerning generalization.

Keywords: Federated Learning Strategies, Relational-Regularized Autoencoder, Time-Series Classification

1. Introduction

The digitalization of industrial processes, marketing and controlling of energy fluxes as well as health care is accompanied by high-frequency data generation on edge devices. This data contains a lot of information about processes that could be optimized and automated. Since there exists a lack of data-privacy and -security as well as a communication latency when operating in the cloud, many companies prefer edge-computing. This governance often means that each company or device has access to a relatively small amount of data, leading to overfitting while applying machine learning models to solve some specific problems. As many applications benefit from sharing their knowledge, *FL* helps to circumvent this conflict by training locally and aggregating model weights on a central server in a loop. *FL* has already been applied to predict load and renewable energy profiles as well as electric vehicle charge demand in a smart grid [6], [14], [11], [12]. Digital twins are incorporated into edge networks to fill the gap between physical systems and digital spaces as well as to improve communication efficiency [8]. Electric vehicles consumption during various traffic and environmental scenarios has been modelled in a co-simulation with a *FL*-framework to model battery behaviour [7]. In future, various new devices like smart meters will be installed locally generating large volume time-series data. Since there are no consistent labels for unique time-series classes between these devices, this paper focuses on the unsupervised time-series classification task to calculate strongest similarities and mutual relations. This could later be used within a local energy system, recognizing time-series relations automatically to optimize its generation in operating mode. In our approach, a *RAE* (see section 2.6) is used to create time-series specific encodings considering their means and variances (see sections 2.4 and 2.6). These encodings could be additionally used to exchange the state of local energy consumption for incorporation into a swarm based approach. Time-series correlation as well as entropy measures do not make sense here since

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 September 2022 doi:10.20944/preprints202209.0435.v1

time-series from devices located far apart possess completely different characteristics in the same time interval. In case of heterogeneous data from different participants on the clients, *FL* can lead to slow convergence and overfiting [2]. Based on the time-series classification task, this paper presents a suitable approach (see section 2.5) to overcome these difficulties.

Related Work

The state of the art *FL* method uses iterative linear weight averaging, based on the number of data-points on each client, to train deep neural networks [9] (see equation 6). This approach shows robust results considering different *dd* and *dq*. Based on this initial work, a lot of research has been done on the issue of heterogeneous data distributions across clients. Federated Adaptive Weighting [18] compares local and global model gradients to develop a non-linear mapping function for weight averaging based on client contribution. This accelerates convergence while excluding non-participating clients. [20] propose the earths mover's distance to measure model weight divergence of different *dd* for weight averaging on the central server. However this requires a small randomly chosen subset from each client on the central server, which violates data-privacy. [15] use client updates to perform active device selection, which speeds up convergence and minimizes the number of communication rounds of federated training. Another option to accelerate convergence is to exclude clients with irrelevant model updates [17]. Additionally the weight drift on various clients can be controlled by using a stochastic averaging algorithm to reduce the effect of different *dd*, improving global model performance at the cost of slower convergence [5]. Besides learning only a single global model, there is additionally the option to learn multiple models [10]. Here the nodes are clustered according to their various *dd* and those with less correlation to their corresponding cluster members or with slow convergence are excluded. This direct exclusion of some clients may be unacceptable in the case of time-series classification tasks. Optimizing multiple models via *FL* using the multi-task learning framework proposed by [13] only works for convex models like support vector machines. Multiple model approaches are not applicable to our usecase since the objective is to learn a global model that should be able to compare many time-series at once. This paper aims to develop a simple alternative to the previously mentioned weight-averaging strategies with respect to the time-series classification task. Assuming weights should be averaged according to the various client *dd* and their relative position to each other (see figure 1), cluster-variance-weighting (*cvw*), client-centroid-weighting (*ccw*) and ensemble-weighting (*ew*) are introduced in section 2.5. They are applied on the encodings learned by the *RAE* (see section 2.6). To analyze these strategies, a scenario is defined in section 2.1. Pre- and post-processing of the data is detailed in section 2.3 and 2.4. The evaluation procedure is explained in section 2.7, followed by a discussion of the results in section 3.

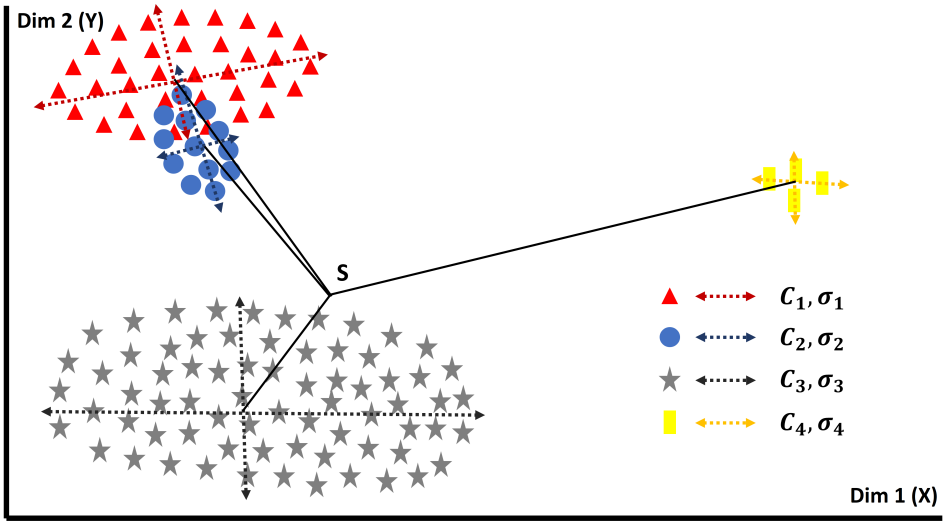


Figure 1. *dd* σ in and centroid *S* between each client *C*

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 September 2022 doi:10.20944/preprints202209.0435.v1

2. Materials and Methods

2.1. Data

A real-valued energy time-series dataset with a quarter-hourly resolution is extracted from [1]. It contains 22 classes of several German grid operators and European countries within a period of time between 2 and 6 years. Each time-series is split into annual intervals to generate multiple sequences per class. This leads to a larger sample set for evaluation statistics. The number of time-series class labels (*tscl*) are as follows:

- Biomass (30)
- Brown coal (18)
- Export (126)
- Fossil gas (30)
- Grid load (30)
- Hard coal (30)
- Hydropower (30)
- Import (126)
- Nuclear energy (18)
- Other conventional (25)
- Other renewables (30)
- Pump storage (30)
- Photovoltaic (30)
- Physical netto-export (26)
- Power price (8)
- Price (83)
- Pump storage (30)
- Quantity called (8)
- Quantity held (10)
- Residual load (30)
- Wind offshore (12)
- Wind onshore (30)
- Working price (8)

The *dd* of various *tscl* can be analyzed by calculating d_{mean} which equals to the mean standard deviation of its encodings (see equation 1, 5 and table 1):

$$d_{mean} = \frac{1}{L}\sum_{j=1}^{L}\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\overrightarrow{enc}_{ts_{i,j}} - \mu_{\overrightarrow{enc}_{ts_{i,j}}})^2} \tag{1}$$

where:

L: number of indices per encoding
N: number of encodings of one *tscl*

Table 1: d_{mean} of each *tscl*

tscl	Variance	Distribution label
Quantity called	0.0324	small
Residual load	0.0538	small
Physical netto-export	0.0652	small
Pump storage	0.0687	small
Power price	0.0704	small

Continued on next page

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 September 2022 doi:10.20944/preprints202209.0435.v1

Table 1: d_{mean} of each *tscl* (Continued)

tscl	Variance	Distribution label
Photovoltaic	0.0706	small
Price	0.0775	small
Grid load	0.0815	medium
Import	0.0816	medium
Wind onshore	0.0874	medium
Fossil gas	0.0904	medium
Export	0.1073	medium
Hard Coal	0.1132	medium
Quantity held	0.1205	medium
Working price	0.125	medium
Wind offshore	0.1305	big
Other conventional	0.1331	big
Hydropower	0.1442	big
Other renewables	0.1538	big
Brown coal	0.1653	big
Biomass	0.1715	big
Nuclear energy	0.1923	big

2.2. Generating Synthetic Datasets

For evaluating different weighting-strategies (see section 2.5) of the trained models, they are evaluated on 10 clients with diverse *dd*. In the following, a scenario is detailed with regard to *dd*:

2.2.1. Evaluation Dataset

While a *RAE* is trained within a *FL*-framework, its evaluation concerning model performance is essential after each *FL*-epoch. Therefore, 20% of randomly chosen time-series of each class (see section 2.1) are extracted from the entire dataset according to various *dd*. These multiple evaluation-datasets located on the central server help to control the number of epochs or even abort the entire process.

2.2.2. Training Dataset

To divide time-series into and generate different *dd*, the central question is "what is a highly distributed time-series". This task is mostly solved by applying statistics or in particular entropy measurements [4]. Unfortunately, these methods only consider the entire distribution but not really the type of sub-sequence of a specific time-series. To Overcome this shortcoming, the method described in section 2.3 and the *RAE* are used to generate daily encodings of Min and Max. By calculating the mean (the mean sub-sequence) and standard deviation (the distribution around the mean sub-sequence) of these encodings, the time-series can be clustered according to their similarity and distribution. The clustering works as follows:

Firstly, the *RAE* is trained on the entire dataset to generate representative encodings $\overrightarrow{enc}_{ts}$ (see section 2.4) of each annual time-series. Having an encodings matrix, *k-means* clustering with $k = 10$ is used to assign each annual time-series to a specific cluster. Based on this pool of different clusters, diverse synthetic client-datasets representing different distributions

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 September 2022 doi:10.20944/preprints202209.0435.v1

and number of datapoints are created for the *FL* setting (see table 2):

Table 2: *FL* setting concerning time-series cluster (number of annual time-series of cluster) with various *dd* (lowest on C1 and highest on C10) on different clients

Client Number	Time-Series Cluster (N Time-Series)
C1	1 (60)
C2	2, 3 (56)
C3	2 ,3, 4 (51)
C4	1, 4, 5, 6 (48)
C5	1 ,2 ,3 ,4 ,5 (45)
C6	2, 3, 4, 6, 7, 8 (42)
C7	1, 2, 3, 4, 5, 6, 7 (35)
C8	1, 2, 4, 5, 6, 7, 8, 9 (24)
C9	1, 2, 3, 4, 5, 6, 7, 8, 9 (18)
C10	1, 2, 3, 4, 5, 6, 7, 8, 9, 10 (10)

2.3. Pre-Processing

Time-series are subject to various exogenous variables and possess different characteristics concerning trend, seasonality and noise. Assuming a characteristic sub-sequence beginning at a local minimum (Min) or maximum (Max), it might be possible to classify distinct time-series. Therefore, Min and Max are chosen on a daily basis, producing a vector with length of an entire day with 96 values for the classification task. The resulting sub-sequences $\overrightarrow{x}_{\text{sub}}$ are then normalized using the monthly maximum of their corresponding time-series $\overrightarrow{x}_{\text{month,max}}$. Monthly normalization is chosen due to the monthly evaluation (see section 2.7):

$$\overrightarrow{x}'_{sub} = \frac{\overrightarrow{x}_{sub}}{\overrightarrow{x}_{month,max}} \tag{2}$$

2.4. Post-Processing

Applying the *RAE* on the pre-processed dataset generates a huge number of encodings. To make these comparable to each other, each encoding has to be standardised by:

$$enc'_{i,j} = \frac{enc_{i,j} - \frac{1}{n}\sum_{i=0}^{n} enc_{i,j}}{\sqrt{\frac{1}{n}\sum_{i=0}^{n}(enc_{i,j} - \mu_{enc,j})^2}} \tag{3}$$

where:

n: total number of encodings
i: i-th encoding
j: index of the encoding
enc: sub-sequence encoding

Each time-series is then characterized by statistics of its Min and Max encodings. At this point, the mean $\mu_{ts,j}$ as well as the standard deviation $\sigma_{ts,j}$ per time-series, year, Min and Max (see section 2.3) are calculated:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 September 2022 doi:10.20944/preprints202209.0435.v1

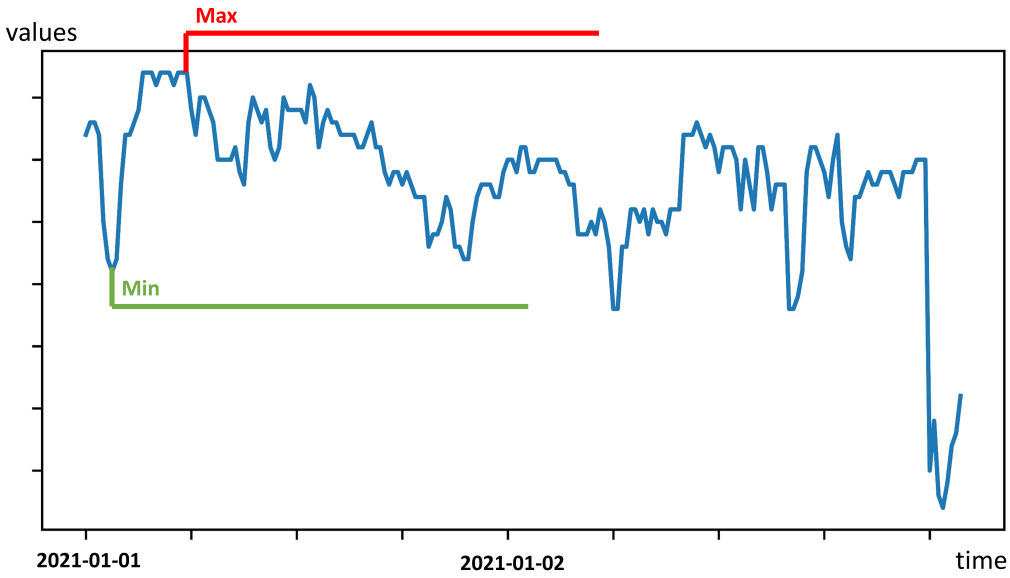


Figure 2. Minimum and maximum sequence of a single day

$$\mu_{,j} = \frac{1}{n}\sum_{i=0}^{n} x_{i,j}; \qquad \sigma_{,j} = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(x_j - \mu_{ts,j})^2} \tag{4}$$

where:

n: number of encodings of one time-series, year, Min or Max

To calculate similarities between different time-series according to the Euclidean distance, the vectors of $\overrightarrow{\mu}_{ts,min}$, $\overrightarrow{\mu}_{ts,max}$, $\overrightarrow{\sigma}_{ts,min}$ and $\overrightarrow{\sigma}_{ts,max}$ are concatenated to include both dimensions as well as its Min and Max characteristics:

$$\overrightarrow{enc}_{ts} = concatenate\left(\overrightarrow{\mu}_{ts,min}, \overrightarrow{\mu}_{ts,max}, \overrightarrow{\sigma}_{ts,min}, \overrightarrow{\sigma}_{ts,max}\right) \tag{5}$$

2.5. Federated Learning Strategies

In classical *FL*, the different client-model-weights (*cmw*) are averaged according to the number of their datapoints [9] (see equation 6). This can be a good method for homogeneous distributed data, but somewhat less for the heterogeneous case. For example, a model trained on a client with a lot of similar data can result in overfitted and non optimal weights. In contrast, training a model on a client with highly diverse data can result in better weights. Thus the latter clients should contribute more when averaging all *cmw* to obtain the best global-model-weights (*gmw*).

$$gmw^{k+1} = \frac{1}{n_{cl}}\sum_{i=1}^{n_{cl}} \frac{a_i^k}{\sum_{i=1}^{n_{cl}} a_i^k} * cmw_i^k \tag{6}$$

where:

- gmw: global model weights
- n_{cl}: number of clients
- a: number of datapoints per client

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 September 2022 doi:10.20944/preprints202209.0435.v1

2.5.1. Cluster variance weighting

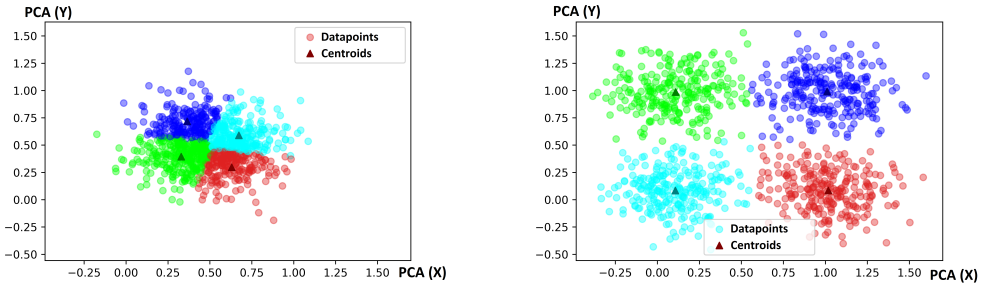
The first weighting strategy applies *k-means clustering* on the encoding matrix of all $\overrightarrow{x}'_{sub}$ on each client. The number of clusters may be variable, but is considered as constant $n_c = 7$. Assuming that *dd* correlates with the variance between the cluster-centers (*cc*), one clients' *cvw* is defined as:

$$cvw = \frac{1}{n_c}\sum_{j=0}^{n_c}\sqrt{\frac{1}{l}\sum_{i=0}^{l}(cc_{i,j} - \frac{1}{l}\sum_{i=0}^{l}cc_{i,j})^2} \tag{7}$$

where:
- l: length of $\overrightarrow{x}_{\text{sub}}$
- *cc*: cluster centers

This distribution score *cvw* now enables *cmw* averaging according to the client distributions:

$$gmw^{k+1} = \frac{1}{n_{cl}}\sum_{i=1}^{n_{cl}} cvw_i^k * cmw_i^k \tag{8}$$



(a) low data-distribution **(b)** high data-distribution

Figure 3. Different client *dd*

2.5.2. Client centroid weighting

While the previous strategy only weights by the intra-client *dd*, it does not handle the inter-client *dd* (see figure 1). Assuming each client's data to be a single time-series, equation 5 becomes:

$$\overrightarrow{enc}_{cl} = concatenate\left(\overrightarrow{\mu}_{cl,min}, \overrightarrow{\mu}_{cl,max}, \overrightarrow{\sigma}_{cl,min}, \overrightarrow{\sigma}_{cl,max}\right) \tag{9}$$

where:
- *cl*: client

Having a mean encoding of all clients which additionally does not fit the length of *RAE*-encodings to reproduce it on the central server, each client can be weighted with respect to the distance to its centroid:

$$ccw_i = \left|\overrightarrow{enc}_{cl,i} - \frac{1}{n}\sum_{k=1}^{n}\overrightarrow{enc}_{cl,k}\right| \tag{10}$$

where:
- *ccw*: client centroid weight

Resulting in *gmw*:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 September 2022 doi:10.20944/preprints202209.0435.v1

$$gmw^{k+1} = \frac{1}{n_{cl}} \sum_{i=1}^{n_{cl}} ccw_i^k * cmw_i^k \tag{11}$$

2.5.3. Ensemble weighting

Since one single strategy is unlikely to perform the best, all previous strategies are combined into an *ew*:

$$gmw^{k+1} = \frac{1}{n_{cl}} \sum_{i=1}^{n_{cl}} \left(\frac{a_i^k}{\sum_{i=1}^{n_{cl}} a_i^k} + \frac{cvw_i^k}{\sum_{i=1}^{n_{cl}} cvw_i^k} + \frac{ccw_i^k}{\sum_{i=1}^{n_{cl}} ccw_i^k} \right) * cmw_i^k \tag{12}$$

2.6. Model

Conventional *Autoencoders* do not regularize the latent space which often leads to non-plausible encodings. In consequence, the encodings are not comparable and are unusable for unsupervised time-series classification. Therefor, this paper applies a *RAE* to overcome this issue [19]. The *RAE* uses a siamese network architecture to regularize the encodings by adding a special loss function $loss_{rel}$ (see equation 13) with regularization factor α to the reconstruction loss $loss_{mse}$ to calculate the total loss *Loss* (see equation 14).

$$loss_{rel} = |\Delta^2_{input} - \Delta^2_{enc}| \tag{13}$$

$$Loss = loss_{mse} + \alpha * loss_{rel} \tag{14}$$

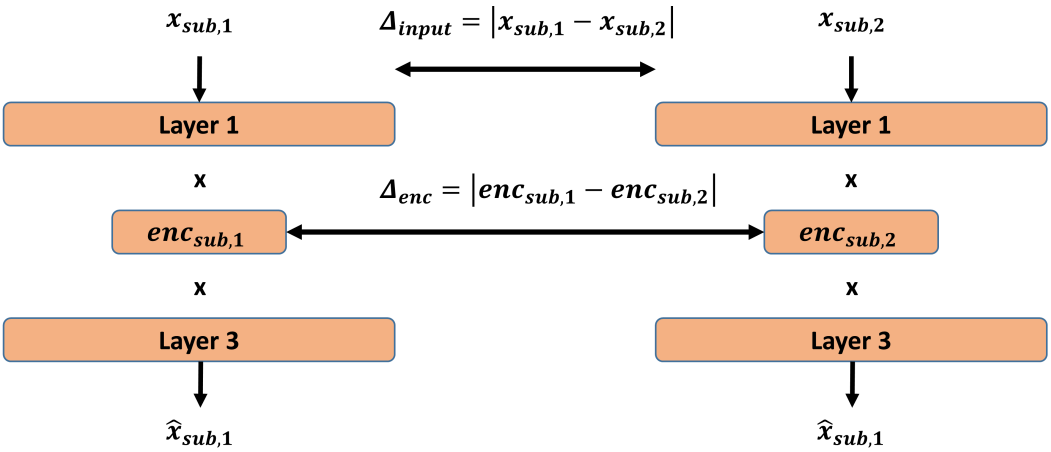


Figure 4. Relational Autoencoder

2.7. Evaluation

Assuming a time-series can be summarized by a distribution of different sub-sequence characteristics, their means $\overrightarrow{\mu}_{ts,min}$, $\mu_{ts,max}$ and standard deviations $\overrightarrow{\sigma}_{ts,min}$, $\overrightarrow{\sigma}_{ts,max}$ are calculated for 3 consecutive months and concatenated to $\overrightarrow{enc}_{ts}$ (see section 2.4). In the next step, the nearest neighbors of each $\overrightarrow{enc}_{ts}$ are calculated using the Euclidean distance. Then a precision score can be calculated by comparing the correct label to the label given by the nearest neighbors of $\overrightarrow{enc}_{ts}$ (see equation 15). Besides this raw classification score, the model loss as well as its convergence time are evaluated.

$$prec_{score,tscl_i} = \frac{1}{N} \sum_{i=1}^{N} hit_i \tag{15}$$

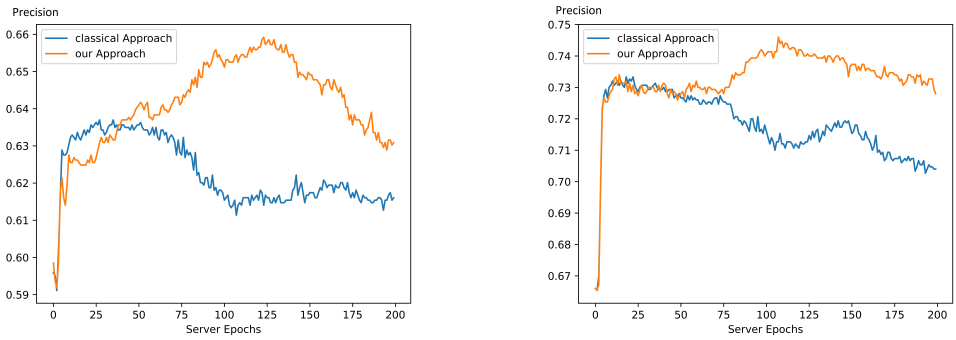
where:

- *hit*: equals to 0, if $tscl_c \neq tscl_{matching}$; else hit = 1
- $tscl_c$: ground truth *tscl*
- $tscl_{matching}$: matched *tscl* by nearest neighbor method

 doi:10.20944/preprints202209.0435.v1

3. Results and Discussion

Applying *FL* with the *ew* strategy shows good results (see figures 5, **??**) in the time-series classification task with respect to the precision score (see section 2.7). While the classical approach (see equation 6) converges faster, *ew* performs better in the long run (up to 120 epochs) and results in better model generalization. Additionally, the results are very sensitive to the evaluation datasets whereas *ew* obviously shows improvements concerning robustness. While the evaluation datasets do not represent the real *dd* on the various clients, the results can only show relative comparisons between the classical approach and *ew*.



(a) Evaluation data A **(b)** Evaluation data B

Figure 5. Precision (y-axis) of *RAE* trained on *FL* setting (see table 2) for various random evaluation datasets (see section 2.2.1), number of epochs (x-axis) and strategy *ew*

4. Conclusion

This paper describes a strategy for classifying time-series using their latent representations. The *ew* strategy is a robust approach for weight averaging and model generalization. Future work will further investigate time-series classification and its parameterization within *FL* considering shapelets [16] and data pre-processing to use it for the next steps in the energy supply chain for energy prediction and energy system optimization. This task also includes the investigation of *Federated Feature-Selection* [3] to analyze the inter-dependency of different parties inside a decentralized energy system.

5. Acknowledgements

The work was financially supported by BMWi in Germany (Bundesministeriums für Bildung und Forschung) under the project "Bauhaus.MobilityLab".

References

[1] URL: https://www.smard.de/en/downloadcenter/download-market-data (visited on 02/16/2022).

[2] Alex Barros et al. "A strategy to the reduction of communication overhead and overfitting in Federated Learning". In: *Anais do XXVI Workshop de Gerência e Operação de Redes e Serviços*. Uberlândia: SBC, 2021, pp. 1–13. DOI: 10.5753/wgrs.2021.17181. URL: https://sol.sbc.org.br/index.php/wgrs/article/view/17181.

[3] Pietro Cassarà, Alberto Gotta, and Lorenzo Valerio. *Federated Feature Selection for Cyber-Physical Systems of Systems*. 2021. DOI: 10.48550/ARXIV.2109.11323. URL: https://arxiv.org/abs/2109.11323.

[4] David Cuesta-Frau. "Slope Entropy: A New Time Series Complexity Estimator Based on Both Symbolic Patterns and Amplitude Information". In: *Entropy* 21.12 (2019). ISSN: 1099-4300. DOI: 10.3390/e21121167. URL: https://www.mdpi.com/1099-4300/21/12/1167.

[5] Sai Praneeth Karimireddy et al. *SCAFFOLD: Stochastic Controlled Averaging for Federated Learning*. 2021. arXiv: 1910.06378 [cs.LG].

[6] Haizhou Liu et al. *A Federated Learning Framework for Smart Grids: Securing Power Traces in Collaborative Learning*. 2021. arXiv: 2103.11870 [cs.LG].

[7] Mingming Liu. *Fed-BEV: A Federated Learning Framework for Modelling Energy Consumption of Battery Electric Vehicles*. Aug. 2021.

[8] Yunlong Lu et al. "Communication-Efficient Federated Learning for Digital Twin Edge Networks in Industrial IoT". In: *IEEE Transactions on Industrial Informatics* PP (July 2020), pp. 1–1. DOI: 10.1109/TII.2020.3010798.

[9] Brendan McMahan et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1273–1282. URL: https://proceedings.mlr.press/v54/mcmahan17a.html.

[10] Xiaomin Ouyang et al. "ClusterFL: A Similarity-Aware Federated Learning System for Human Activity Recognition". In: *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*. MobiSys '21. Virtual Event, Wisconsin: Association for Computing Machinery, 2021, 54–66. ISBN: 9781450384438. DOI: 10.1145/3458864.3467681. URL: https://doi.org/10.1145/3458864.3467681.

[11] Yuris Saputra et al. "Energy Demand Prediction with Federated Learning for Electric Vehicle Networks". In: Dec. 2019, pp. 1–6. DOI: 10.1109/GLOBECOM38437.2019.9013587.

[12] Marco Savi and Fabrizio Olivadese. "Short-Term Energy Consumption Forecasting at the Edge: A Federated Learning Approach". In: *IEEE Access* 9 (2021), pp. 95949–95969. DOI: 10.1109/ACCESS.2021.3094089.

[13] Virginia Smith et al. "Federated Multi-Task Learning". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, 4427–4437. ISBN: 9781510860964.

[14] Afaf Taïk and Soumaya Cherkaoui. "Electrical Load Forecasting Using Edge Computing and Federated Learning". In: June 2020, pp. 1–6. DOI: 10.1109/ICC40277.2020.9148937.

[15] Hao Wang et al. "Optimizing Federated Learning on Non-IID Data with Reinforcement Learning". In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 2020, pp. 1698–1707. DOI: 10.1109/INFOCOM41043.2020.9155494.

[16] Kun Wang et al. "Time Series Classification via Enhanced Temporal Representation Learning". In: *2021 IEEE 6th International Conference on Big Data Analytics (ICBDA)*. 2021, pp. 188–192. DOI: 10.1109/ICBDA51983.2021.9403177.

[17] Luping WANG, Wei WANG, and Bo LI. "CMFL: Mitigating Communication Overhead for Federated Learning". In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. 2019, pp. 954–964. DOI: 10.1109/ICDCS.2019.00099.

[18] Hongda Wu and Ping Wang. *Fast-Convergent Federated Learning with Adaptive Weighting*. 2021. arXiv: 2012.00661 [cs.LG].

[19] Hongteng Xu et al. *Learning Autoencoders with Relational Regularization*. 2020. arXiv: 2002.02913 [cs.LG].

[20] Yue Zhao et al. *Federated Learning with Non-IID Data*. 2018. arXiv: 1806.00582 [cs.LG].