

# Federated learning for scam classification in small Indonesian language dataset: an initial study

Michael Chen, Dareen Kusuma Halim

Department of Computer Engineering, Faculty of Engineering and Informatics, Universitas Multimedia Nusantara, Tangerang, Indonesia

---

## Article Info

### Article history:

Received Sep 23, 2022

Revised Oct 18, 2022

Accepted Nov 18, 2022

---

### Keywords:

Federated learning

Machine learning

Natural language processing

Scam classification

Small dataset

---

## ABSTRACT

Most digital phishing or scam trick users into fraudulent links and is more effective against users with low technology literacy, like in Indonesia. Machine learning is widely used for scam classification, but most require sending the messages to a centralized server. This induces privacy concern as messages might contain private data. Federated learning (FL) was proposed to allow user devices to train models locally without sending data to server. In this work, we examined the use of FL with gated recurrent unit (GRU) model for classifying scam messages in Indonesian language with small dataset. We provided two FL-based baseline models (FedAvg and daisy-chained algorithms) and a dataset for scam classification in Indonesian language. We examined the models based on these performance metrics; precision, recall, F1, selectivity, and balanced accuracy. Despite the performance, we pointed out characteristics of the FL algorithms and the hyperparameters for this use case as pointers towards fine-tuning these baseline models. Overall, the FL model with FedAvg algorithm performed better in all metrics except recall.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



---

## Corresponding Author:

Dareen Kusuma Halim

Department of Computer Engineering, Faculty of Engineering and Informatics

Universitas Multimedia Nusantara

Street of Scientia Boulevard, Gading Serpong, Tangerang 15810, Indonesia

Email: dareen.halim@umn.ac.id

---

## 1. INTRODUCTION

As technology gets more advanced and more integrated into our daily lives, we are seeing more computerized processes generating digital data and storing them either locally or in the cloud. Digital data provides convenience of access and processing but is more prone to breaching. Common causes of digital data breaches are i) account hacking by brute forcing passwords, ii) security holes in the system that could be exploited manually or by malware, and iii) social engineering typically through phishing or scam messages [1]. While account hacking and security holes can be alleviated by system robustness, i.e., keeping out the attacker, social engineering requires more effort to solve as it largely involves ‘human interaction’. Digital phishing or scam messages are commonly encountered through emails, SMS, or other messaging, tricking users into fraudulent links with intention to steal their accounts, installing malwares, or transferring some amount of money to the attackers.

Machine learning has been widely used for detecting and classifying scam messages [2]-[7], but most of them assume a centralized server for processing the datasets. In a real-world setup, this requires the users to send their messages (either from emails, SMS, or other messaging services) to the server, exposing both scam and private messages. These private messages might be sensitive information like payment or login confirmation. Federated learning (FL) [8], [9] was proposed to deal with this privacy issue by having user devices to send only the model parameters to the server.

In a typical FL architecture, a central server coordinates with user clients (devices) that locally trains machine learning models with local data and sends the trained parameters to the server [10]. Federated learning is first proposed in [8], with a widely used FedAvg algorithm where on each training iteration, a number of clients ( $C$ ) perform  $E$  number of local forward pass before sending the result to a central server. The server aggregates results from clients with weighted average function, then sends the weight update to the clients. In this algorithm,  $C$  and  $E$  are controlled parameters.

A survey on use of FL in the natural language processing (NLP) field is done in [11]. The survey reviews the FL algorithms used in various NLP tasks, along with their limitations and challenges. In general, FL-based NLP models still lacks performance-wise compared to centralized ones. In next-word prediction task, long short-term memory (LSTM) is widely used in most works [12]-[14], and gated recurrent unit (GRU) is also employed [15] due to its lower complexity which is more suitable for client mobile devices. GRU has also found similar performance to LSTM in other applications that use historical data [16].

In text classification, two apparent challenges in applying FL are the proper algorithm (e.g., using FedAvg [17]) for aggregating clients locally trained models, and the size of the models [11]. Text classification models like BERT [18] performs well but is too large for client devices. InclusiveFL [19] proposed a method for assigning different-sized models to clients with different computing resources, with the goal of having a big model at the central server.

Although limited, there are several libraries and frameworks that support FL development, like PySyft, Federated AI Technology Enabler, and TensorFlow Federated [20]. A comparison of FL versus centralized machine learning (ML) is done in [20], studying the effect of amount of data, number of nodes, and data distribution on the performance. For the same number of nodes, larger amount of data improves the FL performance, close to that of the centralized ML. Inversely, increasing the number of nodes in FL does not improve the performance, but instead decrease the performance it for large node counts as it reduces amount of data per node. For dealing with small datasets, [21] proposed FedDC method which combines federated averaging and daisy-chaining of local training between clients. The daisy-chaining aspect allows the model to learn from small, distributed datasets. Compared to plain federated averaging and naïve daisy-chaining, FedDC performs better in terms of accuracy but requires higher communication overhead.

In this work, we studied the use of FL in binary classification for scam messages on Indonesian language datasets, due to two main reasons. First, scam messages are more effective against unsuspecting users, typically in countries like Indonesia where technology is still penetrating wider societies, with lower average levels of education or technology literacy. Second, to our best knowledge, there are not many works on scam message classification for Indonesian language, which leads to a very small number of workable datasets. It is particularly so with federated learning models. Our contributions hence are as follows.

- We provided baseline models for binary classification of scam messages in Indonesian language with FL and small datasets. We studied two FL algorithms; federated averaging, and naïve daisy-chaining [21].
- We provide a small dataset for scam message classification in Indonesian language, sourced from SMS messages. We chose SMS as in Indonesia, they are more likely to contain scam or phishing. This dataset is a combination of an internet-provided dataset [22] and a self-collected dataset.

The rest of the paper is organized as follows. Section 2 describes our research method; the dataset used in this work, its distribution and pre-processing, as well as our federated learning models and training parameters. In section 3, we analyze the model performance, intended to serve as a baseline for federated learning models with similar use case and dataset constraints. Finally, section 4 concludes this work.

## 2. METHOD

In this section we describe the Indonesian language dataset used, and the pre-processing performed prior to feeding it into the model. The dataset along with the source code is publicly available in our repository, detailed in acknowledgement section. We also discuss the model's architecture used in this work.

### 2.1. Dataset description

The dataset is a combination of a publicly available dataset in [22] and a self-collected dataset with manual labelling. Each data point is an SMS message, regardless of whether they belong to the same conversation or not. That is, an SMS conversation of 5 messages is seen as 5 separate messages or data points (we did not consider contextuality). Due to this data collection nature, we expect our models to classify on an individual message basis.

Our dataset is small and imbalanced, i.e., 1,780 messages, where 591 are scams (positive class), and 1,189 are normal and promotional messages (negative class). As shown in Table 1, the main data set of 1,780 is into 56% for the training set, 14% for the validation set, and 30% for the testing set. We kept a similar positive and negative class distribution for all sets.

Table 1. Distribution of training and test data

Usage	Scam	Non-scam	Sum by usage
Training + validation	397	843	1240
Test	194	346	540
Total			1780

**2.2. Data format and pre-processing**

The dataset is stored as a creating shared value (CSV) file with two columns, ‘message/text’ and ‘label’. Our GRU model expects a fixed-length message input, hence we perform these data pre-processing steps below (which is also depicted in Figure 1. We kept the original dataset intact, and only perform these pre-processing steps in the memory.

- Lower case and symbol removal: Turning all letters into lower cases, and removing these special symbols “!@#%&\*()” as well as numbers.
- Indonesian stop words removal: Removing known Indonesian stop words that are generally not relevant for the training, based on the list defined in the Scikit-learn library.
- Words tokenization: The message is tokenized into words, i.e., we split them by white spaces. Each word is then converted into a number that represents the index of that word in a previously created bag of words. The bag of words was created from scratch, based on our whole dataset in which we apply the same data pre-processing.
- Padding and truncation: This is done based on an input size threshold (word count). If a message’s word count (tokenized) is less than the threshold, it is pre-padded with zeros. If a message’s word count exceeds the threshold, it is truncated. The output is then fed into the model. In this work, we used a threshold of 30 words. Figure 2 illustrates this process.

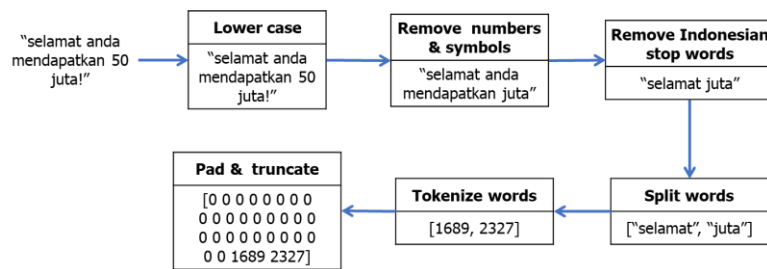
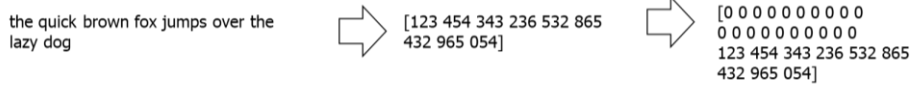


Figure 1. Data pre-processing flow

**Padding**



**Truncation**

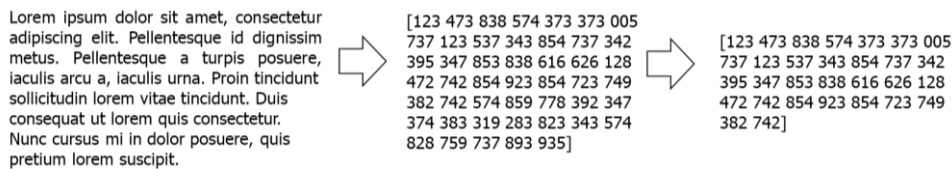


Figure 2. Pad and truncate process examples

**2.3. Model description**

We studied two federated learning algorithms in this work: i) aggregated FL (using FedAvg), and ii) daisy-chained FL (naïve daisy chaining). Due to the small dataset, we only used two virtual workers in the FL setups, referred to as Anne and Bob for convenience. We used horizontal data partitioning [23] for distributing to the workers. The two virtual workers are assigned slightly different positive and negative class distributions in the training data, to emulate such possibility in the real case [24]. That is, Anne had 35% positive and 65% negative classes distribution, while Bob had 28% positive and 72% negative classes

distribution. Table 2 summarizes the distribution of the training data. Due to the difference of training mechanisms in the two setups, workers in the daisy-chained FL were assigned validation data, while workers in the aggregated FL setup were not.

Table 2. Distribution of data on virtual workers

Data holder	(Aggregated FL)		Eval		Sum by data holder
	Scam	Non-scam	Scam	Non-scam	
Anne	173	323	0	0	496
Bob	141	355	0	0	496
Global model	-	-	83	165	248
(Daisy-chained FL)					
Anne	173	323	37	87	620
Bob	141	355	46	78	620
Global model	-	-	0	0	0

#### 2.4. Aggregated and daisy-chained FL algorithm

In the aggregated FL, the training validation is performed on the aggregated (global) model in each iteration. That is, the virtual workers Anne and Bob are trained on their own data set, but the training loss is computed on the global model. The global model is then updated and sent back to the workers for the next iteration. On the other hand, the daisy-chained FL does not aggregate into a global model. Instead, the workers alternately train and validate their local model and local dataset in each iteration and pass the model to the next worker. At the end of the last iteration, the global model is attained. Figure 3 depicts two FL configurations.

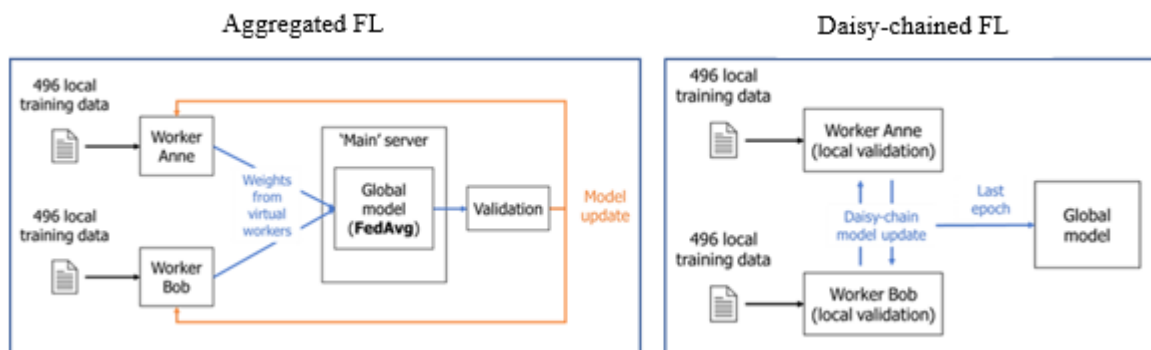


Figure 3. Aggregated FL and daisy-chained FL configuration

#### 2.5. GRU model description, training hyperparameters, and training setup

Both of our virtual workers run a GRU model (no pre-training) based on a handcrafted model [25] by the developer of PySyft, a Python-based federated learning framework used in this work. The GRU architecture is as follows.

- An embedding layer, with an input dimension equals to our self-made vocabulary size, and 50 output neurons.
- A single layer GRU with 10 output neurons.
- A dropout layer for dropping 20% of processed information.
- A dense layer with one output neuron, followed by sigmoid activation.

The binary classification threshold is set dynamically using threshold moving [26]. We used binary cross entropy as the loss function and stochastic gradient descent (SGD) optimizer for both aggregated FL and daisy-chained FL. The training was run with batch size of 30, variations of 100 and 200 epochs, and learning rate of 0.1, 0.01, 0.001. In total, we have six combinations based on the number of epochs and learning rates. The models were trained locally with Jupyter Notebook running on a Dell Vostro 14-3468, Intel i5-7200U processor, 12GB of RAM and ~400GB of disk spaces. Details of the Python environment and required libraries can be found in our code repository, provided in the acknowledgement section [27].

### 3. RESULTS AND DISCUSSION

This work is intended as an initial study for the use of federated learning in scam message classification with a small Indonesian language dataset. Hence, instead of focusing on the raw performance, in this work we focus our evaluation on:

- providing baseline models and performance for federated learning for similar use case (binary classification on small Indonesian language dataset).
- providing insights on the use of and dataset limitation based on analyzed patterns of the results.

For each combination of epochs and learning rate, we ran the training and testing 5 times. The results were averaged and summarized in Tables 3 and 4. As shown in both tables, the initial models in this work did not perform well, i.e., the metrics are hovering around 50%, similar to random guessing. We used precision, recall, F1-score, selectivity, and balanced accuracy as the metrics in this work.

Table 3. Test results grouped by hyperparameter configurations

Epoch	LR	FL configuration	Precision	Recall/TPR	F1-score	Selectivity/TNR	Balanced Accuracy
100	0.001	Aggregated	38.61%	<b>55.75%</b>	<b>45.54%</b>	50.26%	53.01%
		Daisy	<b>38.92%</b>	49.33%	43.46%	<b>56.71%</b>	<b>53.02%</b>
100	0.01	Aggregated	37.09%	52.12%	43.13%	<b>50.61%</b>	51.36%
		Daisy	<b>38.08%</b>	<b>57.31%</b>	<b>45.69%</b>	48.41%	<b>52.86%</b>
100	0.1	Aggregated	36.67%	43.52%	39.72%	<b>57.69%</b>	50.61%
		Daisy	<b>38.44%</b>	<b>56.27%</b>	<b>45.52%</b>	49.28%	<b>52.77%</b>
200	0.001	Aggregated	<b>38.87%</b>	47.88%	42.69%	<b>57.98%</b>	<b>52.93%</b>
		Daisy	38.09%	<b>52.54%</b>	<b>44.08%</b>	52.10%	52.32%
200	0.01	Aggregated	<b>41.11%</b>	<b>53.16%</b>	<b>45.97%</b>	<b>56.95%</b>	<b>55.05%</b>
		Daisy	37.40%	52.02%	43.46%	51.70%	51.86%
200	0.1	Aggregated	<b>37.74%</b>	47.98%	42.13%	<b>54.09%</b>	<b>51.04%</b>
		Daisy	36.04%	<b>53.68%</b>	<b>43.01%</b>	47.38%	50.53%

Table 4. Test results grouped by FL configurations

Epoch	LR	FL configuration	Precision	Recall/TPR	F1-score	Selectivity/TNR	Balanced Accuracy
100	0.001	Aggregated	38.61%	55.75%	45.54%	50.26%	53.01%
100	0.01		37.09%	52.12%	43.13%	50.61%	51.36%
100	0.1		36.67%	43.52%	39.72%	57.69%	50.61%
200	0.001		38.87%	47.88%	42.69%	<b>57.98%</b>	52.93%
200	0.01		<b>41.11%</b>	53.16%	<b>45.97%</b>	56.95%	<b>55.05%</b>
200	0.1		37.74%	47.98%	42.13%	54.09%	51.04%
100	0.001	Daisy	38.92%	49.33%	43.46%	56.71%	53.02%
100	0.01		38.08%	<b>57.31%</b>	45.69%	48.41%	52.86%
100	0.1		38.44%	56.27%	45.52%	49.28%	52.77%
200	0.001		38.09%	52.54%	44.08%	52.10%	52.32%
200	0.01		37.40%	52.02%	43.46%	51.70%	51.86%
200	0.1		36.04%	53.68%	43.01%	47.38%	50.53%

Table 3 groups the results by hyperparameter combinations to highlight the performance difference between aggregated FL and daisy-chained FL, with the bold texts showing which FL configuration performs better on certain metric. Figure 4 depicts the relative performance difference between the two FL configurations where positive values denote the aggregated FL performing better, and vice versa.

- The precision and balanced accuracy metrics are showing similar behavior where the aggregated FL performs better than the daisy-chained FL when the models were trained over longer epochs of 200. If we directly move messages marked as scam into a ‘scam folder’, higher precision is crucial as we do not want to falsely move important messages into the ‘scam folder’.
- Better recall performance is shown by the daisy-chained FL for 4 of the 6 hyperparameter combinations. In most scam classification case, higher recall is more desirable since we want to capture as much scam message as possible. This is best implemented in a system that shows warning to the users for messages marked as scam, instead of moving them into a ‘scam folder’. The F1-score metric is highly affected by the recall metric, hence showing similar behavior.
- Selectivity of the aggregated FL is much higher than the daisy-chained FL for 5 out of 6 hyperparameter configurations. However, selectivity is less impactful compared to precision and recall since scam messages classification normally focus on finding scam messages instead of normal messages.

Table 4 groups data by the FL configurations to highlight overall performance of the models and hyperparameter combinations. The bold texts denote the pairs of FL configuration and hyperparameter that attained best model performance for each metric. It is shown that the aggregated FL performed better on all

measured metrics, except at *recall* where the daisy-chained FL performed better. The aggregated FL trained on 200 epochs and learning rate of 0.01 provides highest overall performance across metrics.

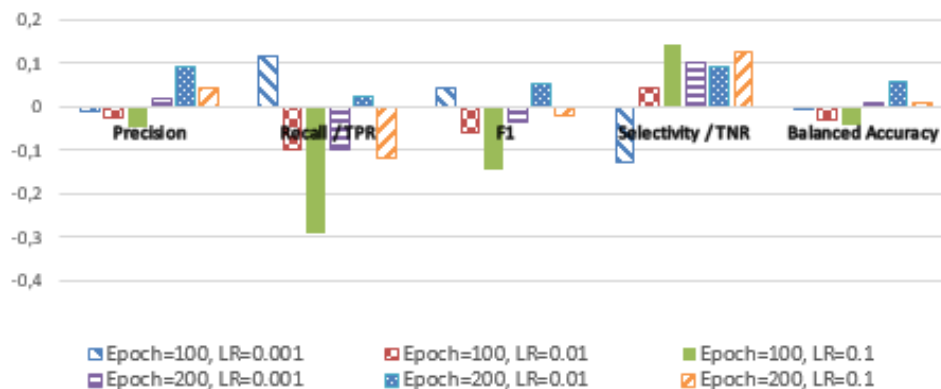


Figure 4. Normalized performance difference of models relative to aggregated FL (Positive value means aggregated FL is better)

#### 4. CONCLUSION

In this work, we examined the use of FL for scam messages classification in Indonesian language with a small dataset over several configurations. We provided two baseline FL models with FedAvg (aggregated FL) and naïve daisy-chained (daisy-chained FL) algorithms, using Python-based PySyft framework and GRU model provided by the Pysyft developers. Despite both FL models performing similarly to random guessing, we found some patterns that could serve as pointers towards fine tuning these baseline models.

The aggregated FL had better precision and balanced accuracy when trained over more iterations. It also attained better selectivity for all hyperparameter configurations except when trained with 100 iterations and learning rate of 0.001. The daisy-chained FL had better recall performance on 4 out of 6 hyperparameter configurations, and in effect the same pattern for F1-score. In terms of best model and configuration for each metric, the aggregated FL attained the best results in precision, F1-score, selectivity, and balanced accuracy. The daisy-chained FL bested the aggregated FL only at the recall performance, trained over 100 iterations and learning rate of 0.01. As there are very few works on using FL for scam message classification in Indonesian language with small dataset, we aimed that our models could serve as the baseline for other future works. We also made our dataset publicly available in our repository for research use, to minimize the lack of such dataset in Indonesian language.

#### ACKNOWLEDGEMENTS

The authors would like to thank Universitas Multimedia Nusantara for supporting this work. The models and dataset used in this work is publicly available in our Github repository.




#### REFERENCES

- [1] Y. Li and Q. Liu, "A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments," *Energy Reports*, vol. 7, pp. 8176–8186, Nov. 2021, doi: 10.1016/j.egy.2021.08.126.
- [2] V. Mhaske-Dhamdhere and S. Vanjale, "A novel approach for phishing emails real time classification using K-means algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 6, pp. 5326–5332, 2018, doi: 10.11591/ijece.v8i6.pp5326-5332.
- [3] M. Raza, N. D. Jayasinghe, and M. M. A. Muslam, "A comprehensive review on email spam classification using machine learning algorithms," in *2021 International Conference on Information Networking (ICOIN)*, Jan. 2021, pp. 327–332, doi: 10.1109/ICOIN50884.2021.9334020.
- [4] T. Jain, P. Garg, N. Chalil, A. Sinha, V. K. Verma, and R. Gupta, "SMS spam classification using machine learning techniques," in *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Jan. 2022, pp. 273–279, doi: 10.1109/Confluence52989.2022.9734128.
- [5] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: review, approaches and open research problems," *Heliyon*, vol. 5, no. 6, p. e01802, Jun. 2019, doi: 10.1016/j.heliyon.2019.e01802.
- [6] S. M. Hameed, "Differential evolution detection models for SMS spam," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 1, p. 596, Feb. 2021, doi: 10.11591/ijece.v11i1.pp596-601.




- [7] A. Rusli, J. C. Young, and N. M. S. Iswari, "Identifying fake news in Indonesian via supervised binary text classification," in *2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, Jul. 2020, pp. 86–90, doi: 10.1109/IAICT50021.2020.9172020.
- [8] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *arXiv e-prints*, Feb. 2016, doi: 10.48550/arXiv.1602.05629.
- [9] M. Asad, A. Moustafa, and T. Ito, "Federated learning versus classical machine learning: A convergence comparison," *arXiv preprint arXiv:2107*, 2020, doi: 10.22541/au.162074596.66890690/v1.
- [10] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140699–140725, 2020, doi: 10.1109/ACCESS.2020.3013541.
- [11] M. Liu, S. Ho, M. Wang, L. Gao, Y. Jin, and H. Zhang, "Federated learning meets natural language processing: A survey," *arXiv preprint arXiv:2107*, Jul. 2021, doi: 10.48550/arXiv.2107.12603.
- [12] S. Ji, S. Pan, G. Long, X. Li, J. Jiang, and Z. Huang, "Learning private neural language modeling with attentive aggregation," in *2019 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2019, pp. 1–8, doi: 10.1109/IJCNN.2019.8852464.
- [13] J. Stremmel and A. Singh, "Pretraining federated text models for next word prediction," in *Future of Information and Communication Conference*, 2021, pp. 477–488, doi: 10.1007/978-3-030-73103-8\_34.
- [14] O. D. Thakkar, S. Ramaswamy, R. Mathews, and F. Beaufays, "Understanding unintended memorization in language models under federated learning," in *Proceedings of the Third Workshop on Privacy in Natural Language Processing*, 2021, pp. 1–10, doi: 10.18653/v1/2021.privatenlp-1.1.
- [15] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," *IEEE Intelligent Systems* 36, Jun. 2019, doi: 10.1109/MIS.2020.3014880.
- [16] S. Hansun, A. Wicaksana, and A. Q. M. Khaliq, "Multivariate cryptocurrency prediction: comparative analysis of three recurrent neural networks approaches," *J Big Data*, vol. 9, no. 1, p. 50, Dec. 2022, doi: 10.1186/s40537-022-00601-7.
- [17] A. Hard *et al.*, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, Nov. 2018, doi: 10.48550/arXiv.1811.03604.
- [18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, Oct. 2018, doi: 10.48550/arXiv.1810.04805.
- [19] R. Liu *et al.*, "No one left behind: Inclusive federated learning over heterogeneous devices," *arXiv preprint arXiv:2202.08036*, Feb. 2022, doi: 10.1145/3534678.3539086.
- [20] A. Budrionis, M. Miara, P. Miara, S. Wilk, and J. G. Bellika, "Benchmarking PySyft federated learning framework on MIMIC-III dataset," *IEEE Access*, vol. 9, pp. 116869–116878, 2021, doi: 10.1109/ACCESS.2021.3105929.
- [21] M. Kamp, J. Fischer, and J. Vreeken, "Federated learning from small datasets," *ArXiv*, pp. 1–13, Oct. 2021, doi: 10.48550/arXiv.2110.03469.
- [22] Y. Wibisono, "Dataset for SMS scam classification in Indonesian language," 2018. [Online]. Available: [http://bit.ly/Yw\\_Sms\\_Spam\\_Indonesia](http://bit.ly/Yw_Sms_Spam_Indonesia).
- [23] Q. Li *et al.*, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Trans Knowl Data Eng*, pp. 1–1, 2021, doi: 10.1109/TKDE.2021.3124599.
- [24] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and S. Avestimehr, "Federated learning for internet of things: Applications, challenges, and opportunities," Nov. 2021, doi: 10.48550/arXiv.2111.07494.
- [25] A. Farias, "Private-AI's federated learning with GRU model," 2019 [Online]. Available: <https://github.com/andreImfarias/Private-AI>.
- [26] C. Esposito, G. A. Landrum, N. Schneider, N. Stiefl, and S. Riniker, "GHOST: Adjusting the decision threshold to handle imbalanced data in machine learning," *J Chem Inf Model*, vol. 61, no. 6, pp. 2623–2640, Jun. 2021, doi: 10.1021/acs.jcim.1c00160.
- [27] M. Chen. "Scam classifier model training using federated learning method," 2022. [Online]. Available: <https://github.com/michaelchen27/scam-classifier>.

## BIOGRAPHIES OF AUTHORS



**Michael Chen**    recently graduated from Universitas Multimedia Nusantara, with a Bachelor of Engineering in 2022. His main interest is in software technology especially mobile devices that utilize hardware sensors and AI powered computing. He is also qualified with HCNA R&S certification in 2020. He can be contacted at email: michael17@student.umn.ac.id.



**Daren Kusuma Halim**    received his bachelor's degree in computer engineering from Universitas Multimedia Nusantara (UMN), Indonesia in 2014, and his Master of Engineering Science (Electrical and Electronics) from Universiti Tunku Abdul Rahman, Malaysia. He was a member of the UTAR VLSI Centre team and together has successfully taped-out a quad-core ARM M0-based MPSoC, the RUMPS401. The chip capability was demonstrated in a complex application of software-defined radio emphasizing on utilizing its low computing resources. He is currently lecturing in Computer Engineering Department, UMN. His research interest includes VLSI and embedded system, distributed system, IoT architecture, and low-resource machine learning. He can be contacted at email: daren.halim@umn.ac.id.