Dean&Francis
ISSN 2959-6157

Federated Learning on Mask recognition with its local simulation and optimization

Liukai Tang

Faculty of engineering, The University of Sydney, Australia
Corresponding author: ltan0578@uni.sydney.edu.au

Abstract:

Even in the post-pandemic era, mask wearing remains widespread. Not only are masks crucial for stopping and managing the spread of illness, but many humanities studies have also shown that people keep wearing masks also because of a variety of psychosocial-behavioural factors. It makes the development of efficient mask recognition technology crucial. Most current research focuses on centralized training. However, especially for mask recognition which involves large-scale data and privacy issues, federated training process has a much larger advantage. Therefore, this paper investigates a basic distributed training of mask recognition model with multi-client participation and central server aggregation. The model is trained on a real-world dataset under multiple model configuration combination (local epoch and client number), then model results like training accuracy, training loss, and training time under different settings are tested under several statistic tests. Finally, this paper explores some balancing strategies regarding local epochs and client numbers, reveals some interactions between two configuration argument (local epoch and client number) and proposes some locally optimal configuration combinations.

Keywords: Federated learning, Distributed system, Mask recognition, Local simulation.

1. Introduction

The importance of mask recognition has been increasingly emphasized in last several years. Especially after the global health crisis caused by the COVID-19 pandemic triggered by the novel coronavirus, masks have emerged as one of the most effective methods for preventing the spread of disease. Additionally, in the post-pandemic era, masks have gradually transitioned from medical protective equipment to an everyday-use thing [1]. Gupta et al.'s research also explored why university students continue to wear masks post-pandemic, focusing on psychological motivations such as emotional control, confidence enhancement, and self-expression concealment [2]. This demonstrates that mask recognition not only holds significant public health value but also plays an important role in studying psychosocial

behaviour. Some studies have also pointed out that even after more than 40 days without local transmission, 62% of people still wear masks in public places. Women, the elderly, and urban areas have higher wearing rates [3]. Therefore, even though mask wearing is no longer as necessary or widespread as during the pandemic, there are still a significant number of people wearing masks, making mask recognition even more necessary.

Large amount of research about mask recognition has been done before. One research has been conducted using deep learning technology to achieve real-time mask detection with a high accuracy, focusing on finding people without a mask in public places [4]. A system based on CNN pre-processing, cropping, and classification can classify three categories (without mask/ incorrectly worn/ correctly worn) in real time and it is applicable to both video streams and images [5]. Also, it is shown that with the combination of Single-Shot Detector (SSD) with MobileNetV2, the average accuracy rate reached 97.8%, with a latency of approximately 0.14 seconds per frame [6]. A Transformer + CNN hybrid model has also been proposed to enhance the ability to capture long-range dependencies in mask detection, achieving an average precision of 89.4% [7]. Additionally, with YOLOv5 and auto-encoders on thermal images, the model can perform mask detection and type classification with *mAP* >97%, making it suitable in conditions without enough light [8].

Most of the previous work seems to focus on centralized training. But for privacy-sensitive mask recognition applications, such as cross-institutional or cross-monitoring network collaborative training, the shift to distributed systems remains crucial. With the aim of attempting to research on the omissions left unaddressed, the author has built a federated learning platform that simulates a real-world environment with multi-client participation and central server aggregation for investigating the impact of different training configurations (e.g. *local_epochs* vs. *num_clients*) on the performance and have performed the result visualization and statistical comparative analysis.

2. Method

2.1 Structure

The whole experiment can be easily triggered by run_all.py. It will iterate through all the configuration combinations (e.g. *local_epochs* vs. *num_clients*) and conduct experiments accordingly. The server and client will be simulated and started as subprocesses. Before the training begins, the images data are evenly distributed by dataset.py among multiple clients to simulate federated data distribution. Every client only uses its local data for multiple epochs of training. Then the server receives models from each client in each round and performs *FedAvg* aggregation and saves *.pt* model file every round. After the entire training process, all model files from each round of training will be saved, and visual training data curve charts and corresponding *CSV* files will also be generated.

The results can be used for final model evaluation on an independent test dataset trough evaluate.py, and statistical testing to verify differences between experimental groups through statistic_test.py, and uploading images to the graphical interface provided in predict_gui.py to detect mask wearing status using the model for practical applications. The basic logic of the workflow is shown in Fig. 1.

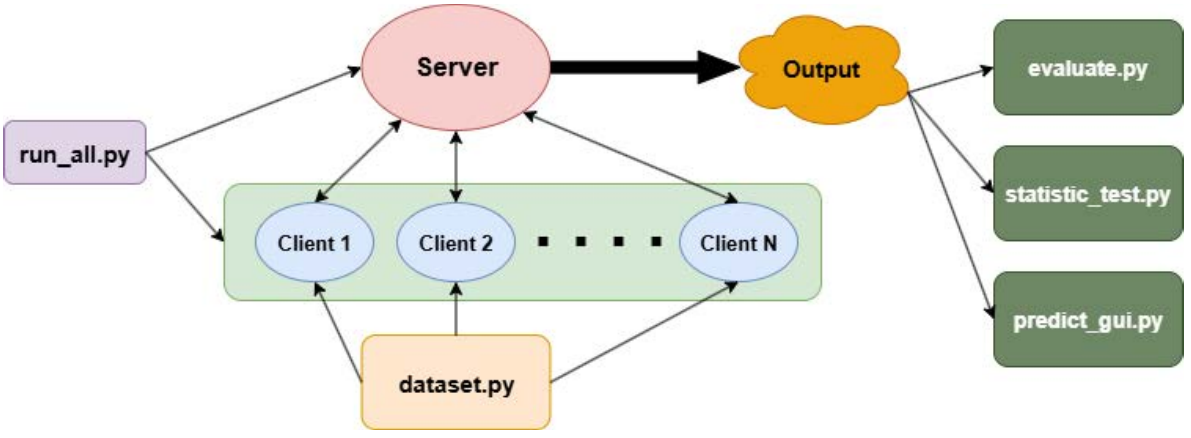

Fig. 1 Project Structure

2.2 Convoluntional neural network model

A custom lightweight convolutional neural network called MaskCNN is used in this experiment to implement binary classification of images of faces with or without mask. The model will take a three-channel color image file with the size of 224*224 as the input. A pair of convolutional

ISSN 2959-6157

layers and a pair of dense layers are included here. The former convolutional layer receives the 3-channel input and uses a 3×3 kernel to output 16 channels, with a 2×2 max-pooling layer following. Similarly, the latter convolutional layer uses a 3×3 kernel and takes 16 input channels to produce 32 channels, also accompanied by another 2×2 max-pooling layer.

The calculation of the size of the convolution layer and pooling layer follows the following formula. The feature map's horizontal and vertical dimensions are *Input*, *K* is the convolution kernel size, *P* is the padding which controls the *Output* size, *S* is the stride which determines the scaling of the *Output* feature map, as shown in formula (1):

$$Output = \left\lfloor \frac{Input - K + 2P}{S} + 1 \right\rfloor. \tag{1}$$

After two rounds of convolution and pooling, the feature map is reduced to a size of 54×54 with 32 channels. This output is then flattened into a one-dimensional vector and passed to the fully connected layers to extract feature and do classification. The first dense layer reduces the 93312 dimensions input to 64 dimensions, with a ReLU activation function. The second dense layer outputs two logits, representing the model's confidence scores for the two classes: wearing a mask and not wearing a mask.

The multi-dimensional tensor is converted into a one-dimensional vector in the flatten layer so that the following fully connected layer can correctly process those data. The formula is shown as in (2), *"combined"* here is just the multiplication of the last three elements from the input.

$$\begin{aligned} &[batch\_size, channel, height, width] \\ &\Rightarrow [batch\_size, combined] \end{aligned}. \tag{2}$$

And the fully connected neural unit is also based on baseline formula in (3). *W* here stands for weight matrix, which is multiplied with input vector *x*, then a bias term *b* is added, and after activation function σ (the author chose ReLu here) is used, the result vector *y* can be calculated:

$$y = \sigma(W \cdot x + b). \tag{3}$$

2.3 Loss function

The CrossEntropyLoss method from PyTorch is chosen as the loss function in this paper. As shown in formula (4), *Z* is the logits, and its subscript indicates his position in the logits, and *y* represent the true label (as a class index), with the value of 0 or 1, and total quantity of different classes is represented by *C*, which is 2 under this paper's certain case.

$$L = -log\left(\frac{e^{z_y}}{\sum_{j=1}^{C} e^{z_j}}\right). \tag{4}$$

In the process of classification, CrossEntropyLoss quantifies the discrepancy between expected class probabilities and actual class label. This method is commonly used when the model outputs raw logits and the ground truth labels are given as class indices and it is not only practical, but also mathematically consistent. Even if the true labels are only category indices, it is still possible to minimize the classification error rate [9].

2.4 Federated Averaging

In this paper, the most fundamental and classic aggregation method, Federated Averaging, is chosen. In this approach, in each round, the server first distributes the global model to some clients, then clients will run multiple rounds of SGD on their own local data, finally at the end of this round, a new global model is generated by the server though just averaging the model arguments sent from clients based on the weighted sample numbers of each clients.

In the formula (5), *w* represents the model argument, *n* stands for the amount of data used by a certain client, and *K* is the total quantity of clients. Default equally weighted average (*n* from all clients are the same), i.e. simple average is used in this study.

$$\bar{\omega} = \frac{1}{\sum_{i=1}^{K} n_i} \sum_{i=1}^{K} n_i \cdot w_i. \tag{5}$$

This method successfully achieves both the speed and flexibility of SGD and the reduction in communication cycles and bandwidth overhead. Also in previous research, FedAvg achieves linear speedup under different conditions like strong convex, convex, and smooth, which indicates that the more devices are involved, the faster convergence occurs, and it still performs well under realistic distributions, which is significantly relevant to the practical issue of mask recognition discussed in this paper [10].

3. Experiment

3.1 Dataset

The model is trained and evaluated on a real-world dataset Real-World Masked Face Dataset, RMFD [11]. This original dataset contains a considerably large number of raw images data of faces with mask and without mask.

Some data-cleaning preprocessing is performed before the experiment. The author randomly selected a certain amount of data from the entire dataset to suit the scale of the experiment, checked the selected image files for duplicate data and removed them to avoid model training bias, and standardized the naming format to ensure that the experimental code could correctly read all data. In the

experiments conducted in this article, 2,000 images for both faces with mask and without mask were used during training phase, while 256 images for both were used during evaluation phase. To prevent overfitting and ensure the authenticity of model evaluation, it is guaranteed that there is no overlap between training data and evaluation data.

If the reader wants to reproduce the experiment with a larger training dataset, the image files should be named and placed in the structure as shown in Fig. 2:

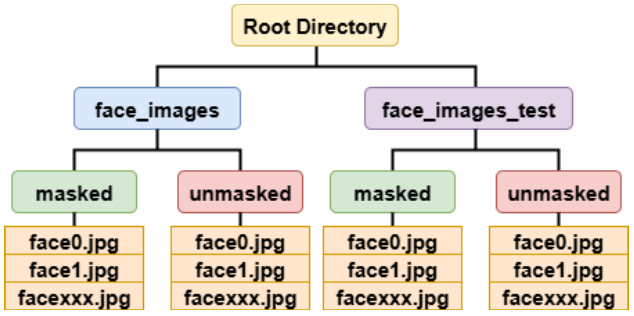

Fig. 2 Training Dataset File Structure

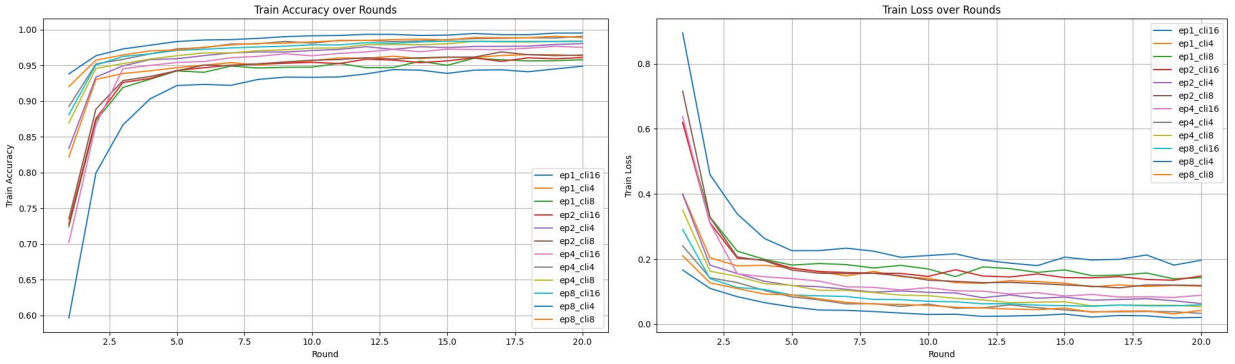

Fig. 3 Training Accuracy and Training Loss Plot

The training accuracy under all configurations increased with each round, indicating that the models had good training capabilities and gradually converged. A rapid increase in accuracy can be seen within the first 5 to 10 rounds for most configurations, and a stable phase follows, indicating that most of the training had been effectively completed in the early stages.

Similarly, under all configurations, the training loss decreases as the number of iterations increases, indicating that the model is continuously learning and fitting the training data. Most curves experience a significant decrease within the first 5 rounds, after which they tend to stabilize, which indicates that the model completes most of the convergence process in the early stages. It is also worth noting that, as shown in the figure, the endpoints of the curves are not as densely clustered as in the left figure but are more dispersed, indicating that the final loss values differ across configurations, which may reflect the actual impact of configuration on model performance.

3.2 Hardware-Settings and System Environment

All training and evaluation processes in this study were conducted on a system equipped with an Intel Core i5-13400F CPU, an NVIDIA GeForce RTX 4070 Ti SUPER GPU, 16 GB of RAM, Windows 10 (22H2), Python 3.12.7, and PyTorch with CUDA 12.1 support.

3.3 Experiment Results

3.3.1 Training Accuracy and Training Loss

In Fig. 3, the two experimental images show the trend of the training accuracy and the training loss respectively of the model as it changes with the number of rounds under different federated learning configurations. Each curve corresponds to a specific combination of local training epochs and the number of clients, labelled as *ep{local_epochs}_cli{num_clients}* in the plot.

3.3.2 Friedman Test on Training Accuracy

Since all configuration combinations show similar trends, focusing on the data at the end of model training is more experimentally meaningful for evaluating the final performance of the model. The author chose to extract the training accuracy from the last 3 rounds. In this study, the Friedman test was used to compare the performance differences of multiple federated learning configurations on the same metric, training accuracy, which is suitable for non-parametric comparisons between multiple related groups. The result is: statistic is 32.7949, and p-value is 0.0006. Since the p-value is far less than the generally accepted significance level, which is usually 0.05, it indicates that there is statistically significant discrepancy in model performance between different configurations, rather than random fluctuations.

ISSN 2959-6157

3.3.3 Nemenyi Test on Training Accuracy

After conducting the Friedman Test to identify whether there are overall significant differences among the tested models or configurations, we proceed to investigate in greater detail which specific groups exhibit statistically significant differences compared to others. This is achieved through the application of the Nemenyi Post-Hoc Test, a widely used pairwise comparison method following non-parametric tests. The results of this test are visualized in Fig. 4, where each cell represents the p-value corresponding to the comparison between two experimental settings. A lower p-value in a cell indicates a stronger statistical difference in performance between the two configurations, suggesting that the observed discrepancy is unlikely to have occurred by chance. The diagonal cells, which compare each method with itself, consistently yield a value of 1, reflecting the absence of any difference in such self-comparisons. Additionally, the figure utilizes a color-coded scheme to intuitively convey the strength of the differences: darker shades represent smaller p-values and thus more significant contrasts, while lighter shades denote weaker or non-significant differences. The accompanying color bar on the right-hand side provides a quantitative reference for interpreting the color intensities across the matrix.

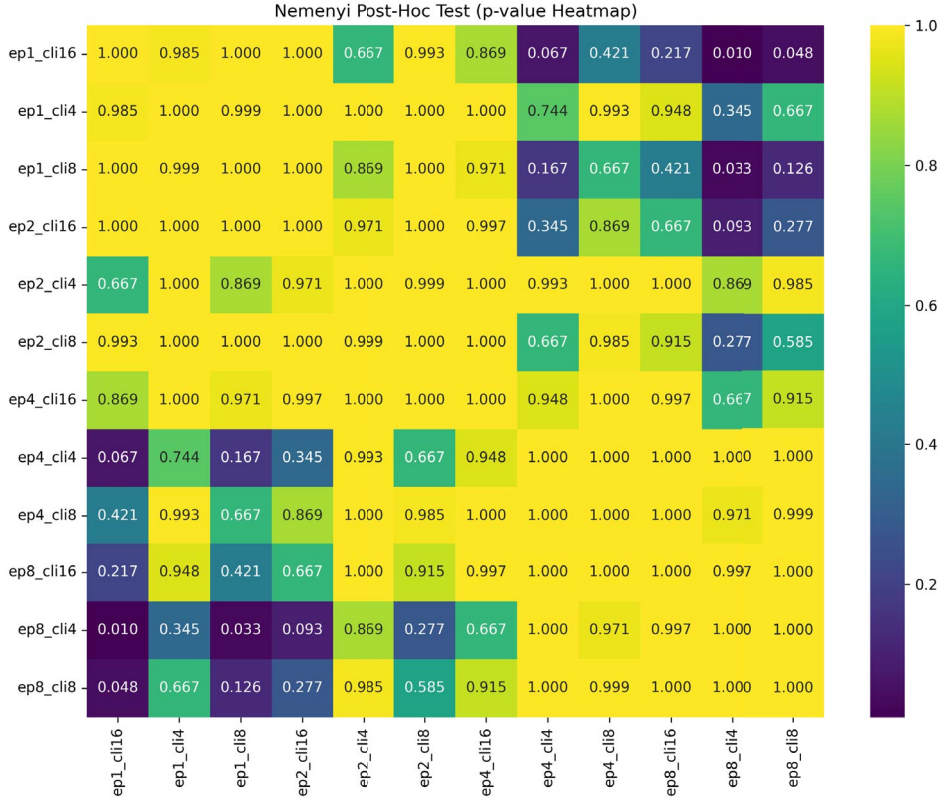

Fig. 4 Nemenyi Post-Hoc Test Plot

The differences between ep8_cli4 and multiple configurations (such as *ep1_cli16* and *ep1_cli8*) are statistically significant. The differences between *ep1_cli16* and *ep8_cli4/cli8* are particularly significant, indicating that there are indeed significant performance differences between poor training configurations and optimal configurations.

The Nemenyi test results confirmed the leading position of the significant performance from configuration group *ep8_cli4* and *ep8_cli8*.

3.3.4 Training Time vs. Average Ranking

In Fig. 5, this Pareto frontier plot illustrates the trade-off between multiple federated learning configurations on two key performance metrics: the horizontal axis represents total training time (in seconds), with lower values being better. The vertical axis represents average accuracy ranking, with lower values indicating higher accuracy level. Through this plot, configurations that achieve the optimal trade-off between training time and accuracy ranking can be easily identified. Grey dots represent the distribution of all configurations across time and accuracy, while red dots connected by red lines form the 'Pareto Frontier points': at these points, no other configuration achieves a better balance across both metrics.

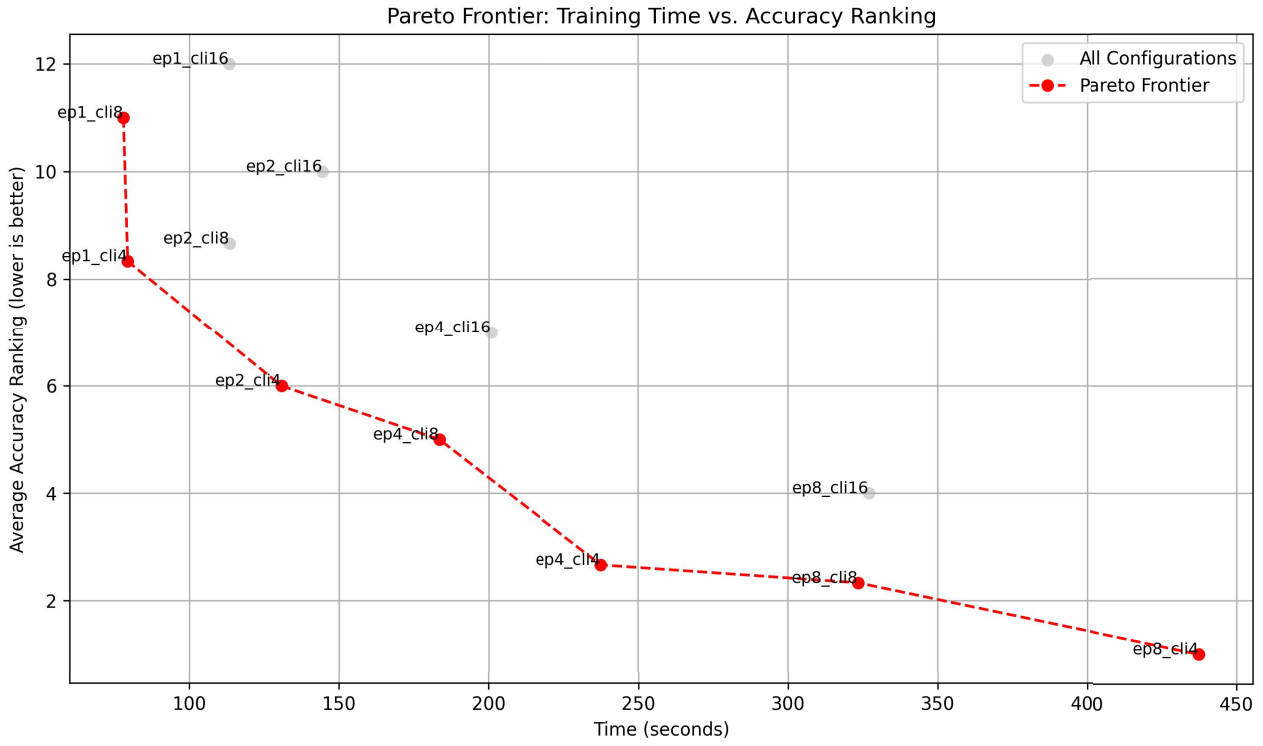

Fig. 5 Pareto Frontier Plot

When resources are abundant, it is recommended to select *ep8_cli4* or *ep8_cli8* to pursue maximum accuracy. Under limited resources, prioritizing choosing *ep2_cli4* or *ep4_cli8* can balance efficiency and performance. Also, configurations that are both slow and have low accuracy, such as *ep1_cli16* or *ep2_cli16* should be avoided using.

3.4 Overall Analysis

The results indicate that the number of local training epochs has a greater impact on model performance; the number of clients affects convergence stability and efficiency; there are significant performance differences, with some configurations significantly outperforming others; and there is a non-linear trade-off between training time and accuracy.

This phenomenon may stem from the role of local epochs: the more local training epochs each client runs, the more completely the model is updated. High-epoch configurations, such as *ep=8*, can better achieve local models' optimization before each round of aggregation, therefore it can improve the convergence quality of the global model. Correspondingly, according to research by Charles et al., local update methods like FedAvg reduce communication requirements and improve convergence efficiency when using more local epochs [12].

The number of clients also has a certain impact on the results. The more clients there are, the more models participate in each round, which theoretically increases data diversity; however, when the number of epochs is small (e.g., *ep=1*), each model is not sufficiently trained, leading to more pronounced oscillations after aggregation. Additionally, more clients can exacerbate aggregation noise, affecting training stability, especially during the initial stages of model training. Also in previous study, it is clearly pointed out that "the variance of the aggregation weights" and "covariance" introduced by client sampling significantly affect the global model's rate of convergence so that having relatively more clients isn't necessarily preferable, and in the case of heterogeneous data distribution, the variance and covariance problems are amplified [13]. This also strongly supports the explanation of why performance degradation occurs in scenarios with multiple clients and fewer epochs (like the ep1_cli16 group in this paper's experiment).

The source code of the project can be checked via the following link: https://github.com/Taneck/Federated-Mask-Recognition

4. Conclusion

In this paper, a federated learning mask recognition system is designed. The model is trained under several model configuration combinations and shows different performance in training accuracy, training loss and training

ISSN 2959-6157

time. A trade-off needs to be considered when choosing a proper model argument. Future research directions could focus on further analyzing the accuracy and generalization ability of the test dataset, introducing non-IID data partitioning for robustness assessment, modelling and evaluating communication costs and energy consumption, and attempting to introduce more complex model structures (such as MobileNetV2) for comparative experiments on configuration sensitivity.

References

[1] Siwei W, Zhao D. Global Design Practice: Mask Design in COVID-19 era. Ergonomics In Design, 2023, 77(77).

[2] Gupta N R, Singh R, Chauhan N, et al. Analyzing the Continuation of Mask-Wearing in the Post-COVID Era: Investigating the Link Between Mask Usage and Self-perception, Self-esteem, and Emotional Expression in Undergraduate Students. Achieving Sustainable Business through AI, Technology Education and Computer Science: Volume 1: Computer Science, Business Sustainability, and Competitive Advantage. 2024: 237-249.

[3] English A S, Li X. Mask use depends on the individual, situation, and location—Even without COVID-19 transmission: An observational study in Shanghai. Frontiers in Psychology, 2021, 12: 754102.

[4] Sethi S, Kathuria M, Kaushik T. Face mask detection using deep learning: An approach to reduce risk of Coronavirus spread. Journal of Biomedical Informatics, 2021, 120: 103848.

[5] Chinnaiyan R, Sai K, Bharath P. Deep learning-based CNN model for classification and detection of individuals wearing face mask. arXiv:2311.10408, 2023.

[6] Sheikh B, Zafar A. RRFMDS: Rapid real-time face mask detection system for effective COVID-19 monitoring. SN Computer Science, 2023, 4: 288.

[7] Al-Sarrar H M, Al-Baity H H. A novel hybrid face mask detection approach using Transformer and convolutional neural network models. PeerJ Computer Science, 2023, 9: e1265.

[8] Kowalczyk N, Sobotka M, Rumiński J. Mask detection and classification in thermal face images. IEEE Access, 2023, 11: 43349-43359.

[9] Mao A, Mohri M, Zhong Y. Cross-entropy loss functions: Theoretical analysis and applications. Proceedings of the International Conference on Machine Learning. PMLR, 2023: 23803-23828.

[10] Qu Z, Lin K, Li Z, Zhou J. Federated learning's blessing: FedAvg has linear speedup. ICLR 2021 - Workshop on Distributed and Private Machine Learning (DPML). 2021.

[11] Wang Z, Huang B, Wang G, et al. Masked face recognition dataset and application. IEEE Transactions on Biometrics, Behavior, and Identity Science, 2023, 5(2): 298-304.

[12] Charles Z, Konečný J. Convergence and accuracy trade-offs in federated learning and meta-learning. International Conference on Artificial Intelligence and Statistics. PMLR, 2021: 2575-2583.

[13] Fraboni Y, Vidal R, Kameni L, Lorenzi M. A general theory for client sampling in federated learning. International Workshop on Trustworthy Federated Learning. Cham: Springer International Publishing, 2022: 46-58.