COMPARING FEDERATED GRADIENT DESCENT AND STOCHASTIC GRADIENT DESCENT FOR MODELING HOUSING VALUE PREDICTIONS

Batuhan Avcı

Aalto University, Espoo, Finland

ABSTRACT

This paper investigates the use of Federated Learning (FL) to improve pricing models in decentralized settings. By utilizing the California Housing dataset, we cluster data based on geographical features to form distinct local datasets, ensuring data privacy. Each cluster corresponds to a local dataset, analogous to real estate agencies in a real-world scenario. We act as a marketplace website, aiding these agencies in their pricing processes. We implement Federated Gradient Descent (FedGD) and Federated Stochastic Gradient Descent (FedSGD) techniques, formulated as a Generalized Total Variation Minimization (GTVMin) problem, to enable efficient parameter sharing among local models. Our results show that FedGD consistently outperforms FedSGD, making it the preferred method in our FL framework. Additionally, our findings reveal that the FL approach significantly surpasses the benchmark linear regression model trained on the entire dataset in terms of performance.

***Index Terms*—** Federated Learning, GTVMin, Personalized Machine Learning, FedGD, FedSGD

1. INTRODUCTION

Federated Learning (FL) is an advanced technique in machine learning that prioritizes data privacy by enabling collaborative model training across decentralized datasets [2, 3, 5]. This approach ensures data remains localized while only sharing model parameters, thus protecting sensitive information [9]. This is particularly vital in fields such as healthcare, business, and climate where private data is used [6, 7]. FL has demonstrated its capability in addressing privacy and data locality issues across various domains, including decision-making for healthcare [14], business [15], and climate [5].

In this paper, we apply FL to the real estate domain, specifically using the California Housing (CH) dataset [11], which includes housing data from various regions across California. Our objective is to predict median house values while maintaining the locality and privacy of the data. To achieve this, we use agglomerative clustering based on geographical features, which results in nine distinct local datasets representing different geographical regions [1]. This clustering simulates a real-world scenario where various real estate agencies handle local data specific to their region. We function as a marketplace website that aids these agencies in their pricing process. FL methods helps us creation of an empirical graph to facilitate parameter sharing between similar datasets [10]. This approach is particularly relevant in the real estate market, where accurate pricing predictions are essential for decision-making, and privacy is critical due to the sensitive nature of the data. We conceptualize an undirected and weighted graph where nodes correspond to local datasets, and edges represent geographical proximity. Nodes with similar datasets are connected, allowing for parameter sharing between them [5]. This setup ensures that the models capture local variations while benefiting from shared knowledge. On top of this, we employ two methods based on Generalized Total Variation Minimization (GTVMin): Federated Gradient Descent (FedGD) and Federated Stochastic Gradient Descent (FedSGD) [4]. These gradient based methods are designed to minimize the variation between local models by penalizing discrepancies between neighboring nodes in the empirical graph [4].

The structure of this paper is as follows. Section 2 defines the problem and explains the creation of local datasets from the CH dataset. Section 3 describes the data preprocessing steps, feature selection process, and the FL algorithms used in our study. Section 4 presents the performance comparison of the FL models alongside the traditional linear regression (LR) model, highlighting the benefits of FL in achieving lower prediction errors and better generalization. Finally, Section 5 summarizes our findings, discusses the implications of our results, and proposes potential areas for future research.

2. PROBLEM FORMULATION

In this section, we clearly define the problem we aim to solve using federated learning with the CH dataset [11]. Our approach involves clustering the data based on geographical features to create distinct local datasets that reflect real-world scenarios.

2.1. Explanation of Local Datasets

We begin with the conceptualization of an empirical graph, depicted as an undirected and weighted graph $G = (V, E)$.

Here, nodes $V := \{1, \ldots, N\}$ correspond to local datasets $D^{(i)}$, where $i \in V$. Each node i in the empirical graph G is tied to an independent local dataset $D^{(i)}$ [4]. For clarity, let's denote a local dataset $D^{(i)}$ as:

$$D^{(i)} = \{(x^{(i,j)}, y^{(i,j)})\}_{j=1}^{m_i} \tag{1}$$

where $x^{(i,j)}$ and $y^{(i,j)}$ represent the feature and label of the j-th data point in the local dataset $D^{(i)}$, respectively, for $j = 1, \ldots, m_i$. The size m_i of the local dataset varies across distinct nodes $i \in V$ [4].

The local datasets were collected from the CH dataset, which provides detailed information about housing characteristics across various regions in the state [11]. The features included in the dataset are listed in Table 1.

Table 1: CH Dataset

Datapoints	Each data point represents a block group in California
Time span	1990 Census
Label	Median house value for households within a block (float)
Features	
longitude	Measure of how far west a house is (float)
latitude	Measure of how far north a house is (float)
housing_median_age	Median age of a house within a block (float)
total_rooms	Total number of rooms within a block (float)
total_bedrooms	Total number of bedrooms for a block (float)
population	Total number of people locating within a block (float)
households	Households within a block (float)
median_income	Households within a block (float)
ocean_proximity	Location w.r.t. ocean/sea (categorical)

2.2. Generation of Local Models and Construction of the Empirical Graph

To adapt the CH dataset for a federated learning setup, we employed agglomerative clustering based on the latitude and longitude features. This clustering process resulted in nine distinct clusters, each representing a geographically coherent subset of the data.

The following features were added to the dataset to facilitate subsequent analysis:

- **central_lat**: Central latitude of each cluster (float).
- **central_lon**: Central longitude of each cluster (float).
- **area_index**: Cluster index (categorical).

Each cluster is treated as a node in the empirical graph $\mathcal{G}$. The central latitude and longitude of each cluster were calculated and assigned as the coordinates for these nodes, ensuring that the graph accurately represents the geographical distribution of the clusters. The clustered areas can be seen in Figure 1.

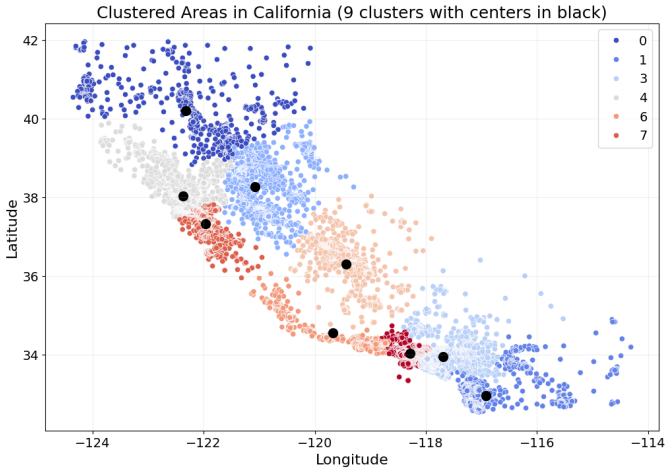

Fig. 1: Clustered areas in California

To construct the empirical graph, we utilized the provided `kneighbors_graph()` method from the `scikit-learn` library [12]. This method connects each node to its nearest neighbors based on the Euclidean distance between their central coordinates. The weights of the edges, denoted by $A_{ii'}$, were determined by the Euclidean distances between the nodes. This method ensures that the edge weights accurately reflect the geographical proximity of the clusters. The resulting graph is a connected, undirected, weighted empirical graph derived from the original nine local data clusters.

2.3. Formulation in the Form of GTVMin

We employ Generalized Total Variation Minimization (GTVMin) for federated learning. The GTVMin problem is generally formulated as:

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w} \in \mathbb{R}^d} \sum_{i \in \mathcal{V}} L_i(\mathbf{w}^{(i)}) + \alpha \sum_{(i,i') \in \mathcal{E}} A_{ii'} \|\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\|_2^2 \tag{2}$$

Here, $L_i(\mathbf{w}^{(i)})$ represents the local loss for node i, $\mathbf{w}^{(i)}$ denotes the local model parameters for node i, α is the regularization parameter, and $A_{ii'}$ are the edge weights in the empirical graph $\mathcal{G}$ [4].

3. METHODS

3.1. Local Data Points

The dataset consists of 20,640 instances, each representing a block group in California. These instances were divided into nine distinct clusters based on their geographical coordinates (latitude and longitude) to create local datasets for each node in our empirical graph. Table 2 shows the number of data points in each cluster.

Table 2: Number of Data Points in Each Cluster

Cluster	Number of Data Points
0	731
1	1,915
2	2,081
3	3,535
4	2,911
5	1,286
6	837
7	2,157
8	5,187

3.2. Feature Selection Process

The feature selection process was meticulously carried out to ensure the inclusion of the most relevant features for our analysis. The following steps were taken:

- **Domain Knowledge**: Features such as geographical coordinates (longitude and latitude), housing characteristics (housing_median_age, total_rooms, total_bedrooms), and demographic information (population, households, median_income, ocean_proximity) were selected based on their relevance to housing prices.
- **Correlation Matrix**: The correlation was significant in our feature selection process. It helped us identify features with strong correlations to the target variable. Although some of the features seems uncorrelated, we decide to use all 16 features for the training.

For normalization and subsequent analysis, the following features were selected: longitude, latitude, housing_median_age, total_rooms, total_bedrooms, population, households, median_income, area_index, central_lat, central_lon, and the one-hot encoded ocean_proximity in 5 categories.

3.3. Data Preprocessing and Normalization

All features, except from ocean_proximity, is normalized using the min-max scaling to facilitate the learning process. Normalization is performed in two ways:

- ***Normalization across the whole dataset:*** This is applied to features area_index, central_lat, and central_lon which are defined with respect to the entire dataset. The Min-Max scaling for a feature x across the entire dataset D is formulated as [13]:

$$x'^{(j)} = \frac{x^{(j)} - \min(x)}{\max(x) - \min(x)} \tag{3}$$

  where $x'^{(j)}$ is the scaled value for the j-th data point, $x^{(j)}$ is the original value, $\min(x)$ is the minimum value of feature x in the entire dataset D, and $\max(x)$ is the maximum value of feature x in the entire dataset D.

- ***Normalization within local datasets:*** This was applied to the remaining features (longitude, latitude, housing_median_age, total_rooms, total_bedrooms, population, households, median_income) within each local dataset $D^{(i)}$. Each local dataset is identified by a unique area_index. The Min-Max scaling for a feature x within a local dataset $D^{(i)}$ is formulated as [13]:

$$x'^{(i,j)} = \frac{x^{(i,j)} - \min(x^{(i)})}{\max(x^{(i)}) - \min(x^{(i)})} \tag{4}$$

  where $x'^{(i,j)}$ is the scaled value for the j-th data point in the local dataset $D^{(i)}$, $x^{(i,j)}$ is the original value, $\min(x^{(i)})$ is the minimum value of feature x in the local dataset $D^{(i)}$, and $\max(x^{(i)})$ is the maximum value of feature x in the local dataset $D^{(i)}$.

To maintain consistency, the target variable, median house value, is scaled down by dividing by 10,000.

3.4. Federated Learning Methods Based on GTV Minimization

We employed two different FL methods based on GTVMin: FedGD and FedSGD [4]. For each local dataset, we choose linear models as the local models. The linear local models are selected due to their interpretability and computational efficiency. This local model selection enables us to compare FedGD and FedSGD models intuitively.

3.4.1. FedGD

The FedGD method updates the local model parameters by solving the GTVMin problem through gradient descent. The update rule for each node i at iteration k is given by [4]:

$$\mathbf{w}^{(i,k+1)} := \eta \Bigg[\frac{2}{m_i} (\mathbf{X}^{(i)})^\top (\mathbf{y}^{(i)} - \mathbf{X}^{(i)} \mathbf{w}^{(i,k)}) + 2\alpha \sum_{i' \in \mathcal{N}(i)} A_{ii'} (\mathbf{w}^{(i')} - \mathbf{w}^{(i)}) \Bigg] + \mathbf{w}^{(i,k)}, \tag{5}$$

where η is the learning rate, $\mathbf{X}^{(i)}$ and $\mathbf{y}^{(i)}$ are the feature matrix and target vector for node i, and $\mathcal{N}(i)$ represents the neighbors of node i in the empirical graph.

3.4.2. FedSGD

The FedSGD method updates the local model parameters using stochastic gradient descent, which processes data in mini-batches. The update rule for each node i at iteration k is given by [4]:

$$\mathbf{w}^{(i,k+1)} := \eta \left[\frac{2}{B} \sum_{r \in B} (\mathbf{x}^{(i,r)})^\top (\mathbf{y}^{(i,r)} - (\mathbf{x}^{(i,r)})^\top \mathbf{w}^{(i,k)}) + 2\alpha \sum_{i' \in \mathcal{N}(i)} A_{ii'} (\mathbf{w}^{(i')} - \mathbf{w}^{(i)}) \right] + \mathbf{w}^{(i,k)}, \tag{6}$$

where B is the batch size, η is the learning rate, and $\mathbf{x}^{(i,r)}$ and $\mathbf{y}^{(i,r)}$ are the feature vector and target value for the r-th data point in the mini-batch for node i.

3.5. Loss Functions for Local Models

Each local model at node i is trained using the L2 loss function, suitable for our regression task involving the median house value. The L2 loss was chosen for its ability to penalize larger errors, which is crucial when minimizing significant prediction errors. Its convexity also simplifies optimization with gradient-based methods. In contrast, L1 loss, while an option, is less sensitive to large errors, making it less suitable for minimizing large deviations. The loss function for the FedGD algorithm is defined as [4]:

$$L_i(\mathbf{w}^{(i)}) = \frac{1}{m_i} \sum_{j=1}^{m_i} (y^{(i,j)} - \mathbf{w}^{(i)T} x^{(i,j)})^2 \tag{7}$$

where $\mathbf{w}^{(i)}$ represents the local model parameters, $x^{(i,j)}$ are the feature vectors, $y^{(i,j)}$ are the corresponding labels, and m_i is the number of data points in the local dataset $D^{(i)}$.

For the FedSGD algorithm, the loss function is computed over a mini-batch of the local dataset, defined as [4]:

$$L_i(\mathbf{w}^{(i)}) = \frac{1}{|\mathcal{B}_t^{(i)}|} \sum_{j \in \mathcal{B}_t^{(i)}} (y^{(i,j)} - \mathbf{w}^{(i)T} x^{(i,j)})^2 \tag{8}$$

where $|\mathcal{B}_t^{(i)}|$ is the size of the mini-batch used at iteration t.

3.6. Variation Measures of Local Models

The variation between local models is measured using the norm of the difference between their parameters. This measure is integrated into the regularization term of the GTVMin formulation, which penalizes discrepancies between models of neighboring nodes. Specifically, for models at nodes i and i' connected by an edge in the empirical graph, the variation measure is $\|\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\|_2^2$. The variation measure ensures that local models trained on geographically proximate data points are similar. This is achieved by minimizing the term $\alpha \sum_{(i,i') \in \mathcal{E}} A_{ii'} \|\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\|_2^2$ [4].

3.7. Construction of Training and Validation Sets

The construction of training and validation sets is carried out through a multi-step process to ensure the robustness of the models. Each local dataset $D^{(i)}$ is shuffled and then split into a training set (60%), a validation set (15%), and a test set (25%).

4. RESULTS

To determine the optimal parameters, we perform a grid search over the hyperparameters on the validation set [8]. For both FedGD and FedSGD, we search over the number of neighbors k in $\{3, 4, 5\}$, the regularization parameter α in $\{0.5, 0.2, 0.1\}$, and the learning rate η in $\{0.1, 0.05, 0.01\}$. For FedSGD, we additionally search over the batch sizes of 16 and 32. The algorithm is run for 1000 iterations to ensure convergence. The optimal parameters are selected based on performance on the validation set and tested on the test set to evaluate the generalization performance. The optimal hyperparameters for both methods are identified as: number of neighbors $k = 3$, regularization parameter $\alpha = 0.1$, and learning rate $\eta = 0.1$. For FedSGD, a batch size of 32 is also used. These parameters were chosen based on the lowest average validation error across all nodes using the Mean Squared Error (MSE) as the evaluation metric.

To benchmark the FL models, we also perform LR on the entire dataset. The results of the LR model are included for comparison. The test sets for each node are constructed by shuffling the local datasets and splitting them into training, validation, and test sets by using percentages of 60-15-25 respectively. Table 3 shows the average training, validation, and test errors for FedGD, FedSGD, and the linear regression model. The results indicate that both FedGD and FedSGD achieve lower errors compared to the LR model.

Table 3: Average MSE for FedGD, FedSGD, and LR

Method	Train	Val	Test
FedGD	43.14	46.37	44.62
FedSGD	43.78	46.78	45.30
LR	50.75	54.51	53.64

Table 4 presents the training, validation, and test errors for each node in the empirical graph for both FedGD and FedSGD.

Table 4: Node-Specific Errors for FedGD and FedSGD

Node	FedGD			FedSGD		
	Train	Val	Test	Train	Val	Test
0	22.79	24.77	22.69	22.85	25.04	22.80
1	43.32	58.45	48.15	43.40	58.00	48.16
2	19.00	22.24	18.98	19.20	22.50	19.17
3	32.69	41.54	34.60	32.79	41.86	34.61
4	64.26	68.24	72.72	65.22	68.58	73.60
5	12.66	13.71	12.64	14.50	15.25	15.09
6	63.54	64.03	62.78	64.26	65.38	63.34
7	57.57	52.34	54.09	57.48	52.39	54.17
8	72.46	71.98	74.96	74.32	71.99	76.73

Figure 2 and 3 presents the results of the both algorithms for 3rd node. The ground truth labels shown in blue and predicted values for test set in orange.

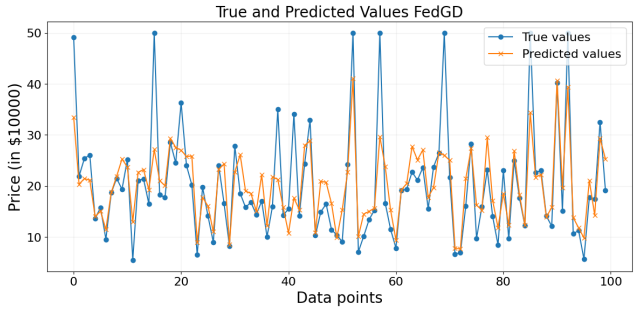

Fig. 2: True and Predicted Values FedGD

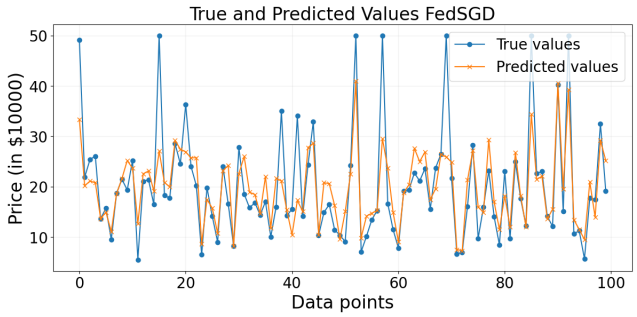

Fig. 3: True and Predicted Values FedSGD

The empirical graph for both FedGD and FedSGD is constructed using the k-nearest neighbors method with $k = 3$. In this graph, an edge between two nodes signifies that they are among each other's three closest neighbors based on geographical coordinates. The edge weights are set to 1 if the nodes are connected. Figure 4 illustrates the empirical graph with $k = 3$.

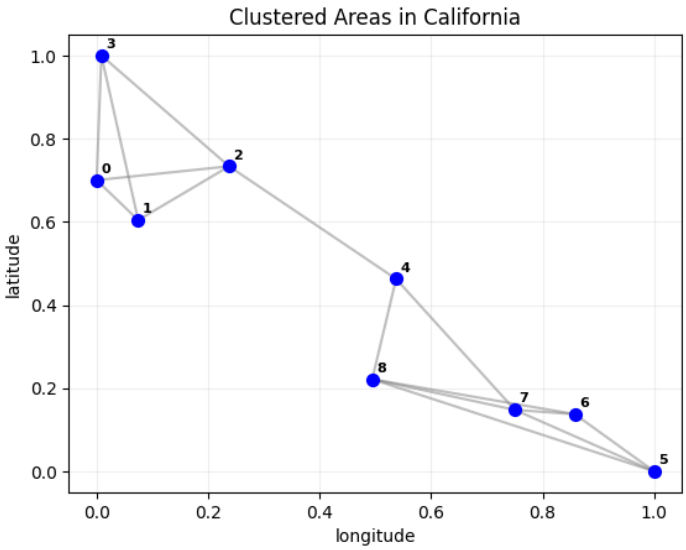

Fig. 4: Empirical graph with 3 nearest neighbors

Analyzing the results, we observe that FedGD consistently achieves lower test errors across most nodes compared to FedSGD. Despite the stochastic nature of FedSGD, which can introduce variability in performance, FedGD demonstrates more stable and reliable outcomes. Therefore, we select FedGD as the final method for our FL framework. The average test errors indicate that FedGD generalizes well to unseen data, achieving slightly better performance overall.

5. CONCLUSION

In this paper, we investigate the use of FL methods for the CH dataset by clustering the data into geographically coherent local datasets and employing GTVMin. The FedGD and FedSGD are formulated as GTVMin instances and used in the framework. The results demonstrated that FedGD consistently achieves lower test MSEs compared to FedSGD, making it the preferred method for our FL framework. Our findings also indicate that the FL approach significantly outperformed the benchmark LR model trained on the entire dataset, highlighting the importance of capturing local variations in data.

However, there are areas for potential improvement. The training errors at nodes 1, 3, and 4 were notably smaller than the corresponding validation errors, suggesting possible overfitting.Even though we expect to see balanced distributions, Nodes 1, 3, and 7 exhibit non-homogeneous distributions for the validation and test sets, as evidenced by the differences in their errors. Additionally, the number of data points in each local model were not balanced, which might have affected the variation minimization process and overall performance. Future work could focus on mitigating overfitting by collecting more training data, especially for nodes with higher errors, or

by exploring different choices for the GTVMin regularization parameter α.

6. REFERENCES

[1] M. R. Ackermann et al. "Analysis of Agglomerative Clustering". In: *Algorithmica* 69 (2014), pp. 184–215.

[2] K. Bonawitz et al. "Towards Federated Learning at Scale: System Design". In: *arXiv preprint arXiv:1902.01046* (2019).

[3] K. A. Bonawitz et al. "Practical Secure Aggregation for Federated Learning on User-Held Data". In: *CoRR* abs/1611.04482 (2016). arXiv:1611.04482. URL: http://arxiv.org/abs/1611.04482.

[4] A. Jung. *Lecture Notes for CS-E4740 Federated Learning*. Available at: https://github.com/alexjungaalto/FederatedLearning/blob/main/material/FL_LectureNotes.pdf. 2024.

[5] A. Jung. *Machine Learning: The Basics*. Singapore: Springer, 2022.

[6] P. Kairouz et al. "Advances and Open Problems in Federated Learning". In: *arXiv preprint arXiv:1912.04977* (2019).

[7] T. Li et al. "Federated Learning: Challenges, Methods, and Future Directions". In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60.

[8] P. Liashchynskyi and P. Liashchynskyi. "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS". In: *CoRR* abs/1912.06059 (2019). arXiv:1912.06059. URL: http://arxiv.org/abs/1912.06059.

[9] B. McMahan et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *Artificial Intelligence and Statistics*. PMLR. 2017, pp. 1273–1282.

[10] H. B. McMahan et al. "Federated Learning of Deep Networks using Model Averaging". In: *CoRR* abs/1602.05629 (2016). arXiv:1602.05629. URL: http://arxiv.org/abs/1602.05629.

[11] R. K. Pace and R. Barry. "Sparse Spatial Autoregressions". In: *Statistics Probability Letters* 33.3 (1997), pp. 291–297.

[12] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[13] D. Singh and B. Singh. "Investigating the Impact of Data Normalization on Classification Performance". In: *Applied Soft Computing* 97 (2020), p. 105524. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2019.105524. URL: https://www.sciencedirect.com/science/article/pii/S1568494619302947.

[14] J. Xu et al. "Federated Learning for Healthcare Informatics". In: *Journal of Healthcare Informatics Research* 5.1 (2021), pp. 1–19.

[15] Q. Yang et al. "Federated Learning: Concept and Applications". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2 (2019), pp. 1–19.