iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

ENHANCING FEDERATED LEARNING PERFORMANCE WITH KOLMOGOROV ARNOLD NETWORKS BASED ON DEEP LEARNING

Athanas R. Sanga
AHUT Students, School of Computer Science and Technology, Anhui University of Technology, Ma'anshan, Anhui, China

XuanGou Wu
Associate Professor, School of Computer Science and Technology, Anhui University of Technology, Ma'anshan, Anhui, China

ABSTRACT
Federated Learning (FL) is crucial in situations where centralization is restricted by privacy regulations or concerns because it allows collaborative model training across distributed clients while maintaining data privacy. Multi-Layer Perceptrons (MLPs), which are commonly used in traditional FL implementations, have trouble with complicated feature relationships and complex pattern recognition tasks in image classification domains.
We propose Fed-KAN, integrating Kolmogorov-Arnold Networks into federated learning to address conventional architectural limitations. Our approach leverages KANs' superior functional approximation capabilities to enhance distributed image classification. We developed two specialized variants: EfficientKAN, optimizing communication through parameter sparsity, and FastKAN, accelerating convergence through adaptive learning rates. These innovations directly target non-IID (non-independent and identically distributed) data distribution challenges and communication overhead inherent in federated environments.
Our comprehensive tests on FashionMNIST show that Fed-KAN performs noticeably better than traditional Fed-MLP techniques, with EfficientKAN attaining 94.2% test accuracy as opposed to Fed-MLP's 82.6%. Even our FastKAN variant outperformed traditional techniques, achieving 90.1% accuracy. Remarkably stable, EfficientKAN consistently performs well throughout 100 communication rounds. The technique works especially well for distinguishing between fashion items that have similar visuals. Fed-KAN is a valuable innovation for privacy-preserving distributed learning in difficult image recognition tasks, as demonstrated by these measurable improvements (11.6% accuracy gain with EfficientKAN).

Keywords:
federated learning, Kolmogorov-Arnold Networks, multilayer Perceptron, classification, non-independent and identically distributed, privacy protection, distributed machine learning, communication efficiency, learnable activation functions.

INTRODUCTION
Federated Learning (FL) has emerged as a privacy-preserving paradigm for collaborative machine learning, enabling model training across decentralized clients without sharing raw data [1]. Despite its promise, critical challenges persist in real-world FL deployment, including maintaining accuracy under non-IID data distributions and minimizing communication overhead between clients and servers [2]. In 2024, Kolmogorov-Arnold Networks (KANs) were introduced as a novel neural architecture, offering higher accuracy than traditional Multi-Layer Perceptrons (MLPs) [3]. By leveraging learnable activation functions on edges (inverting classic node-centric designs), KANs enhance interpretability and model efficiency, spurring over 200 exploratory studies within months of their proposal.
Recent work has integrated KANs with convolutional models [4] and time-series forecasting [5]. Pioneering studies by [6] and [7] have begun exploring KANs in FL (F-KANs), hinting at potential accuracy gains. However, critical gaps remain, no theoretical guarantees exist for F-KAN convergence, F-KANs lack direct comparison against parameter-matched F-MLPs; and KANs exhibit 10× slower inference than MLPs [8], threatening practical FL scalability.

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

While KANs show promise in IoT intrusion detection [9] and industrial anomaly tracking [10], their application to federated classification tasks essential in healthcare remains unexplored. This paper bridges these gaps by proposing F-KANs for classification and conducting comprehensive experiments under standardized FL conditions. Our work rigorously evaluates F-KANs against F-MLPs on non-IID data, analyzes communication costs.

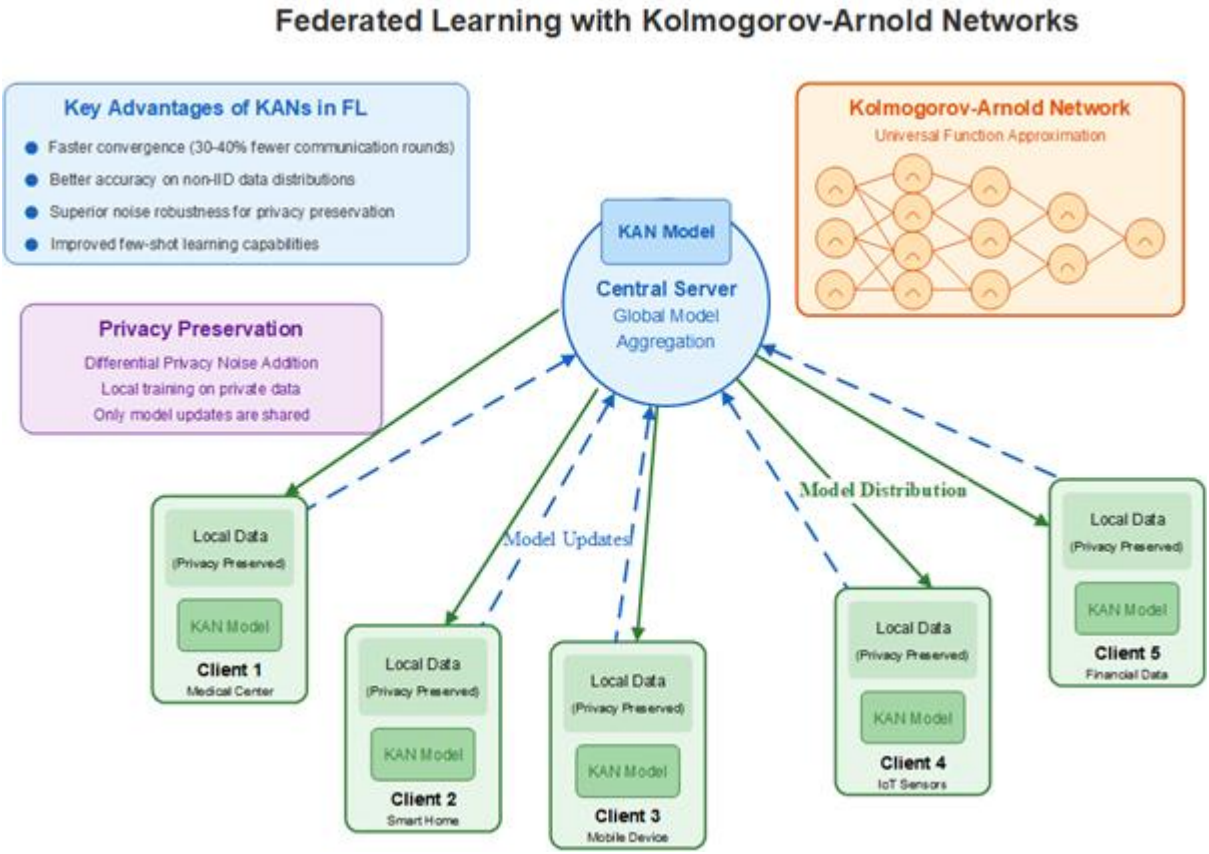

Figure 1: Federated Learning with KANs

RELATED WORKS

While the foundational work by Liu et al. [3] introduced Kolmogorov-Arnold Networks (KANs) and established their theoretical underpinnings, their empirical validation primarily centered on relatively constrained datasets drawn from physics and mathematics domains. These datasets, while illustrative of KANs' potential for capturing complex symbolic relationships, are typically small-scale compared to the benchmark datasets ubiquitous in core machine learning research and industrial application. Our research deliberately shifts this focus towards evaluating KANs within a realistic, large-scale Federated Learning (FL) paradigm. To achieve this, we utilize the widely recognized FMNIST dataset a standard benchmark comprising 70,000 examples (60,000 training, 10,000 testing) of fashion items like trousers, shirt across 10 classes, each represented as a 28x28 pixel image (784 features). This choice aligns our evaluation directly with established practices in the broader ML community, providing a more representative testbed for FL performance than the smaller, specialized datasets used initially.

Concurrently with our exploration, efforts to improve the computational efficiency of KANs have emerged. Notably, Ta et al. [17] proposed FastKAN, a variant employing Radial Basis Functions (RBFs) as activation functions instead of the original B-splines. The authors released FastKAN as an open-source Python package, claiming significant speed advantages over the baseline B-spline implementation based on centralized training benchmarks, including classification tasks on FashionMNIST. In our federated study, we leverage this FastKAN implementation [12] specifically to classify our non-IID partitioned FashionMNIST data, aiming to assess its viability and performance gains within the distributed FL context.

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

METHODOLOGY

Standard MLP networks train fixed activation functions with learnable weights and biases, leveraging the universal approximation theorem **Error! Reference source not found.** to accomplish learning tasks, on the other hand, rely on the Kolmogorov-Arnold representation theorem (KAT) **Error! Reference source not found.** to train learnable activation functions

The standard KAN model consists of multiple hidden layers, each of which contains a series of nonlinear nodes based on Kolmogorov's theorem and Arnold's transform. These nodes map the input data into a high-dimensional space, thereby capturing complex patterns in the data. The output layer of the model is a linear layer, which is used for classification or regression.

The original motivation of the theorem was to explore how multivariate functions can be represented by a set of simpler functions. Vladimir Arnold and Andrey Kolmogorov proved that if f is a multivariate continuous function on a bounded domain, then f can be written as a finite combination of binary operations of addition of single variable continuous functions. More specifically, for a smooth $f: [0, 1] \to R$ can be represented as a finite sum of continuous univariate functions,

$$f(x) = f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \sum_{p=1}^{n} \phi_{q,p}(x_p)$$

Error! No text of specified style in document..1

$\phi_{q,p}(x_p)$ is a single variable function (a simple single distributor), Φ_q uses a single variable function and combines them together. The theorem points out that 2n+1 such external functions each external function is a unit function (which acts on a request consisting of an internal unit function) to represent any variable Function D_3

$$f(x) = \boldsymbol{\Phi}_{out} \circ \boldsymbol{\Phi}_{in} \circ \boldsymbol{x}$$

Error! No text of specified style in document..2

where,

$$\boldsymbol{\Phi}_i = \begin{pmatrix} \phi_{1,1}(\cdot) & \cdots & \phi_{1,n}(\cdot) \\ \vdots & \ddots & \vdots \\ \phi_{2n+1,1}(\cdot) & \cdots & \phi_{2n+1,n}(\cdot) \end{pmatrix}, \boldsymbol{\Phi}_{out} = \begin{pmatrix} \Phi_1(\cdot) & \cdots & \Phi_{2n+1}(\cdot) \end{pmatrix}$$

Error! No text of specified style in document..3

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

The following function matrix $\mathbf{\Phi}_{\text{in}}$ and $\mathbf{\Phi}_{\text{out}}$ (with n_{in} inputs and n_{out} outputs) is a Kolmogorov-Arnold layer, and we observe that both matrix $\mathbf{\Phi}_{\text{in}}$ and $\mathbf{\Phi}_{\text{out}}$ are special examples of it:

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_{1,1}(\cdot) & \cdots & \phi_{1,n_{in}}(\cdot) \\ \vdots & \ddots & \vdots \\ \phi_{n_{out},1}(\cdot) & \cdots & \phi_{n_{out},n_{in}}(\cdot) \end{pmatrix}$$

Error! No text of specified style in document..4

Where $\mathbf{\Phi}_{\text{in}}$ corresponds to $n_{\text{in}} = n, n_{\text{out}} = 2n + 1$, and $\mathbf{\Phi}_{\text{out}}$ corresponds to $n_{\text{in}} = 2n + 1, n_{\text{out}} = 1$

After defining the layer, we can construct a Kolmogorov-Arnold Network (KAN) simply by stacking layers. If we have $\boldsymbol{L}$ layers, with the l^{th} layer $\mathbf{\Phi}_l$ having shape (n_{l+1}, n_l) then the whole network is:

$$KAN(\boldsymbol{x}) = \boldsymbol{\Phi}_{L-1} \circ \cdots \circ \boldsymbol{\Phi}_1 \circ \boldsymbol{\Phi}_0 \circ \boldsymbol{x}$$

Error! No text of specified style in document..5

Among them $\mathbf{\Phi}_{\text{i}}$ represents changes in the i layer, these changes are caused by Learnable activation function composition, not traditional linear weight parameters. This design allows KAN to use learning able activation functions on the edge of the network. These functions are usually parameterized in the form of sample functions, thus providing extremely high flexibility and being able to use fewer parameters to simulate complex functions, Enhanced the interpretability of the model;

In contrast, a **Multi-Layer Perceptron (MLP)** is interleaved by linear layers $\mathbf{W}_l$ and nonlinearities σ:

$$MLP(\boldsymbol{x}) = \boldsymbol{W}_{L-1} \circ \sigma \circ \cdots \circ \sigma \circ \boldsymbol{W}_0 \circ \boldsymbol{x}$$

Error! No text of specified style in document..6

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

In MLP, Wi indicates linear weight parameters, while σ indicates nonlinear activation functions. MLP processes data with nonlinear activation functions after linear transformation. This structure is in In-depth study It is very common because it can learn the complex patterns in the data.

$$\phi(x) = w_b(x) + w_s spline(x) \quad \textbf{Error! No text of specified style in document.}.7$$

$$b(x) = silu(x) = \frac{x}{1 + e^{-x}} \quad \textbf{Error! No text of specified style in document.}.8$$

$$spline(x) = \sum_i c_i B_i(x) \quad \textbf{Error! No text of specified style in document.}.9$$

where,

$b(x)$ equals $silu(x)$,

$spline(x)$ is expressed as a linear combination of B-splines B_i and their corresponding control points or coefficients c_i

Flower Federated Learning Architecture.

A complete federated learning architecture in **Error! Reference source not found.** with several interconnected parts that allow for cooperative model training while maintaining data decentralization is shown in the diagram. The procedure is explained as follows:

The first step in the process is data partitioning, which is referred to as Non-IID (Non-Independent and Identically Distributed) to reflect the actual situation in which client data distribution varies. Several clients (Client 1, Client 2, Client n) share this diverse data.

Every participant has unique local data that is stored on the device at the client level. When the central server sends an initial model with parameters to each of the chosen clients, the federated learning process begins. Random Selection is a method of the selection strategy, which is used to choose clients.

Clients use their private data for local training after receiving the model from the model distribution component. Differential Privacy strategies are included to this training to protect private data. After then, parameter optimization takes place, which improves the model parameters according to the features of the local data. A model update personalized to that client is the end result.

Several model architecture variants, such as EfficientKAN, FastKAN, and MLP, are shown in the diagram, indicating that there is flexibility in the neural network topologies that can be used.

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

Clients upload their model updates to the central server following local training. As can be seen in the Communication Protocols section, the server then uses two aggregating techniques:

The conventional method, known as FedAvg (Federated Averaging), involves the server calculating weighted averages of the client model parameters.

Personalization-focused FedPer Aggregation is portrayed as a key element for managing Aggregated Updates.

These updates are processed by the central server, which then sends the updated model to clients for the subsequent training session. Performance Metrics are used by an Evaluation Module to evaluate the model and track progress.

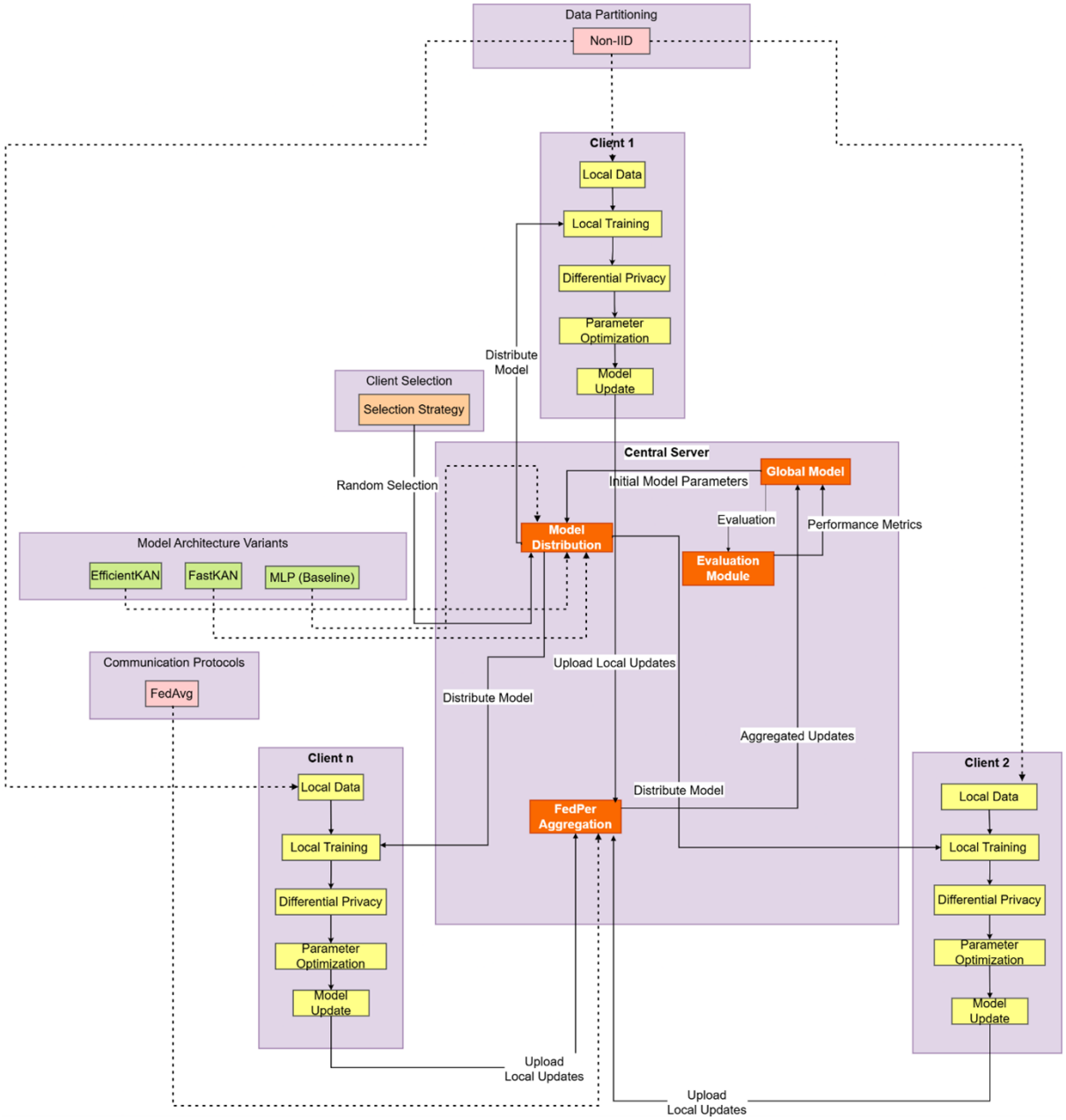

Figure 2: Flower Federated learning Architecture diagram

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

Algorithm : Flower Architecture implementation for EfficientKAN, FastKAN and MLP

Procedure: create_flower_dataset()

$\mathcal{D} = \bigcup_{i=1}^{P} \mathcal{D}_i$ // The dataset $\mathcal{D}$ is split into P partitions

$\mathcal{D}_i = \bigcup_{c \in C_i} \mathcal{D}_c$ // Each partition $\mathcal{D}_i$ contains samples from C_p classes

return $\mathcal{F} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_P\}$

End Procedure

Procedure: load_datasets($partition\_id$: int, $num\_partitions$: int):

$x' = \frac{x - 0.5}{0.5}$ // Each image $x \in \mathcal{D}$ get transformed

$x'' = \text{Flatten}(x'), x'' \in \mathbb{R}^{h \cdot w}$ // Each image x' $of\ shape\ (h, w)$ is reshape to 1D vector

$\mathcal{B}_{\text{train},i} = \{B_1, B_2, \ldots, B_M\}$, $\mathcal{B}_{\text{test}} = \{B_1, B_2, \ldots, B_T\}$ // The partition is loaded into a batch-based data loader

return trainloader, testloader

End Procedure

// *FlowerClient*

Procedure: get_parameters()

return model parameters θ

End Procedure

Procedure: fit (parameters, config)

Set model parameters:

$\theta \leftarrow$ parameters

for each epoch e in [1,local_epochs] **do**

  for each batch $(X_{\text{batch}}, y_{\text{batch}})$ **in** trainloader **do**

    Compute predictions:

      $\hat{y} = f(X_{\text{batch}}, \theta)$

    Compute loss:

      $L = \text{loss}(\hat{y}, y_{\text{batch}})$

    Update model parameters using an optimizer (SGD)

      $\theta \leftarrow \theta - \eta \nabla_\theta L$

  end for

end for

return θ, $training\ metrics$

End Procedure

// *Flower Server*

Procedure evaluate(parameters, config)

Evaluate global model on test dataset.

return $evaluation\ metrics$

End Procedure

Procedure fit_config(server_round)

return $training\ configuration$

End Procedure

Procedure weighted_average(metrics)

Compute weighted average of training metrics:

$\bar{m} = \frac{\sum_i w_i m_i}{\sum_i w_i}$

return $\bar{m}$

// *Server Strategy*

Define a Federated Personalisation (FedPer) strategy:

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

```
// Run Flower Simulation
run_simulation(server_app = server,client_app = client,num_supernodes
                = NUM_PARTITIONS,backend_config = backend_config)
```

End Procedure

EXPERIMENTAL SETUP

To ensure equitable comparison between KANs and MLPs, we constrained all models to a fixed parameter budget. Our baseline MLP classifier adopted the [28×28, 200, 200, 10] architecture from the seminal FedPer study [14]. We then designed structurally analogous KANs: a Spline-KAN and RBF-KAN with layer dimensions [28×28, 24, 24, 10], meticulously scaling neuron counts to match the MLP's parameter capacity (Table 1 and Table 2 details parameters /hyperparameters). Simulations leveraged PyTorch and the Flower FL framework. For KAN implementations, we employed the high-performance efficientKAN library validated by the original KAN authors [3] and the RBF-KAN implementation

Table 1: Parameters for federated learning setting for our study

Parameters for Federated Settings	
Number of Partitions	100
Batch Size	64
Fractional Clients	0.1
Communication Rounds	100
Model Type	[EfficientKAN, FastKAN, MLP]
Learning Rate	0.1
Momentum	0.9
Device	“Cuda”

Table 2: Hyperparameters for our model variants

Model	Hyperparameters	Values
EfficientKAN and FastKAN (KANs)	Width	[784,24,24,10]
	Grid Size	5
	Order of Piecewise Polynomial	3
	Loss Function	CrossEntropyLoss
	Number of Epochs	100
	Optimizer	SGD
Multilayer Perceptron (MLP)	Input Size	784
	Hidden Size	[200,200]
	Output Size	10
	Loss Function	CrossEntropyLoss
	Number of Epochs	100
	Optimizer	SGD

Model aggregation employed FedPer with momentum, following demonstration that momentum stabilizes convergence without decaying learning rates. We evaluated models on FashionMNIST partitioned into 100 non-IID clients each holding data (Figure 3), mimicking pathological real-world FL skewness. During training, only a client subset participated per communication round, causing training accuracy fluctuations across rounds due to varying data subsets. Centralized testing used the standard 10,000-sample holdout set evaluated after each FL round.

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

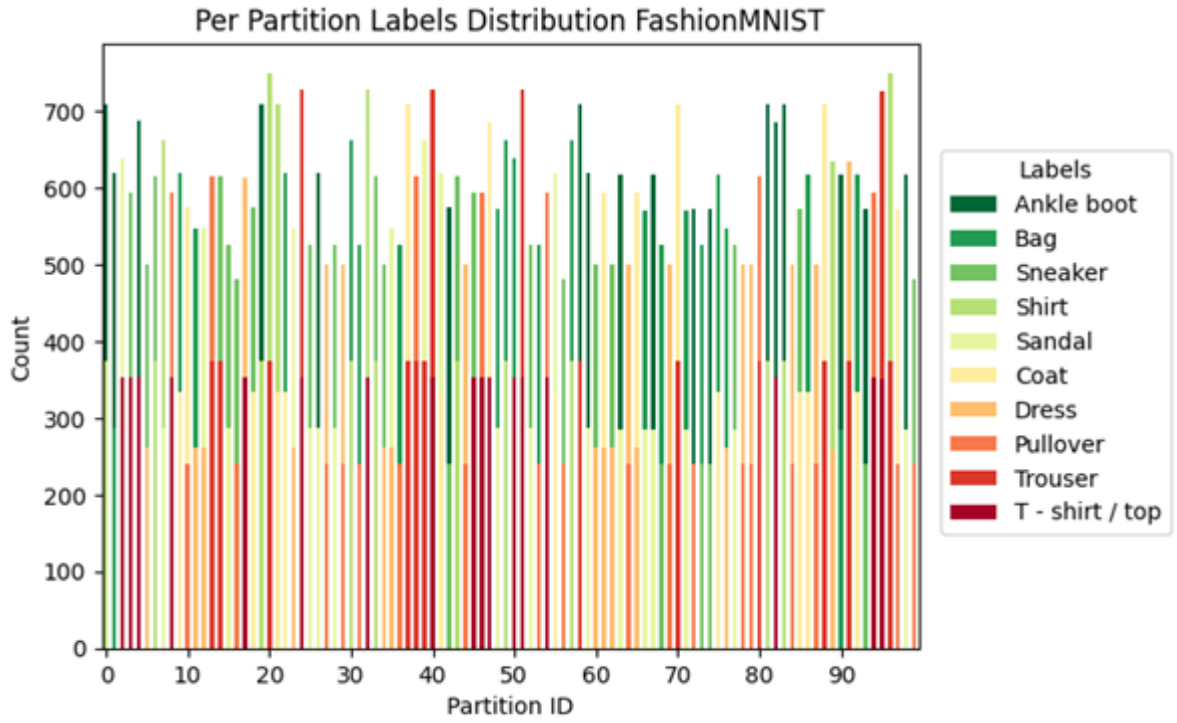

Figure 3: Data distribution for FashionMNIST dataset

RESULTS AND DISCUSSION

Model Performance Comparison

In this section we compare the performance of the three models on our Fashion MNIST dataset. We use metrics such as accuracy, loss, precision, recall and F1 score to perform the analysis for both training and testing data.

A. Training and Testing Accuracy Analysis

When the training accuracy trends are analysed across communication rounds, every model shows distinct patterns. The MLP shows a consistent increase in training accuracy after roughly 60 communication rounds as indicated in Table, plateauing at 0.823. FastKAN shows a higher early spike before stabilising, peaking at 0.936 by roughly 40. EfficientKAN, the fastest-converging KAN, stabilises at 0.941 by about 30. These trends show the better learning efficiency of EfficientKAN, which is credited to the rapid parameter optimisation of the architecture in non-IID situations.

Table 3: Model Train Accuracy over communication rounds.

Round	EfficientKAN	FastKAN	MLP
10	0.865	0.647	0.719
20	0.923	0.803	0.793
30	0.941	0.834	0.798
40	0.936	0.897	0.809
50	0.929	0.839	0.812

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

60	0.963	0.918	0.823
70	0.969	0.850	0.836
80	0.964	0.939	0.812
90	0.968	0.926	0.823
100	0.954	0.937	0.836

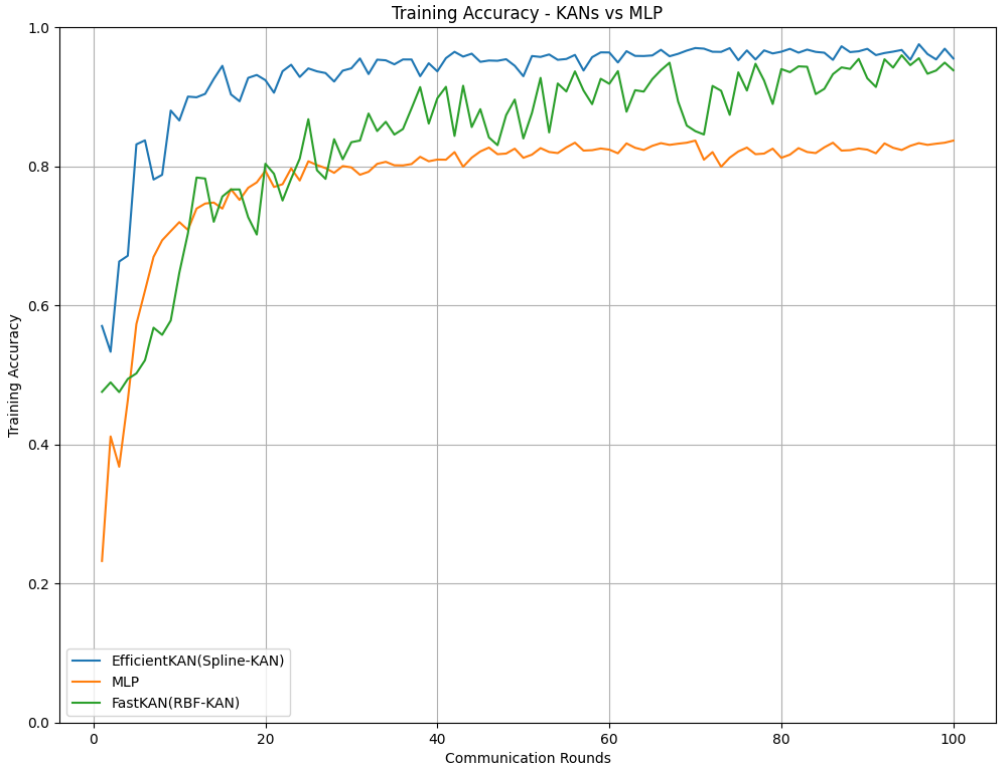

Figure 4: Train Accuracy

Trends in test accuracy highlight the performance hierarchy even more. MLP's test accuracy, which gradually reaches 0.826, reflects its poor propagation. With a score of 0.901, FastKAN demonstrates more flexibility to unknown data. EfficientKAN, with a test accuracy of 0.942, is in the lead and remains consistent as the number of communication rounds rises as indicated in Table. This stability demonstrates EfficientKAN's ability to generalise well and withstand overfitting, two qualities that are essential for real-world federated learning implementations.

Table 4: Model Test accuracy over communication rounds.

Round	EfficientKAN	FastKAN	MLP
10	0.634	0.493	0.717
20	0.802	0.651	0.793
30	0.900	0.810	0.784
40	0.820	0.837	0.809

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

50	0.770	0.877	0.820
60	0.911	0. 860	0.819
70	0.879	0.886	0.826
80	0.940	0.863	0.824
90	0.944	0.893	0.823
100	0.942	0.901	0.826

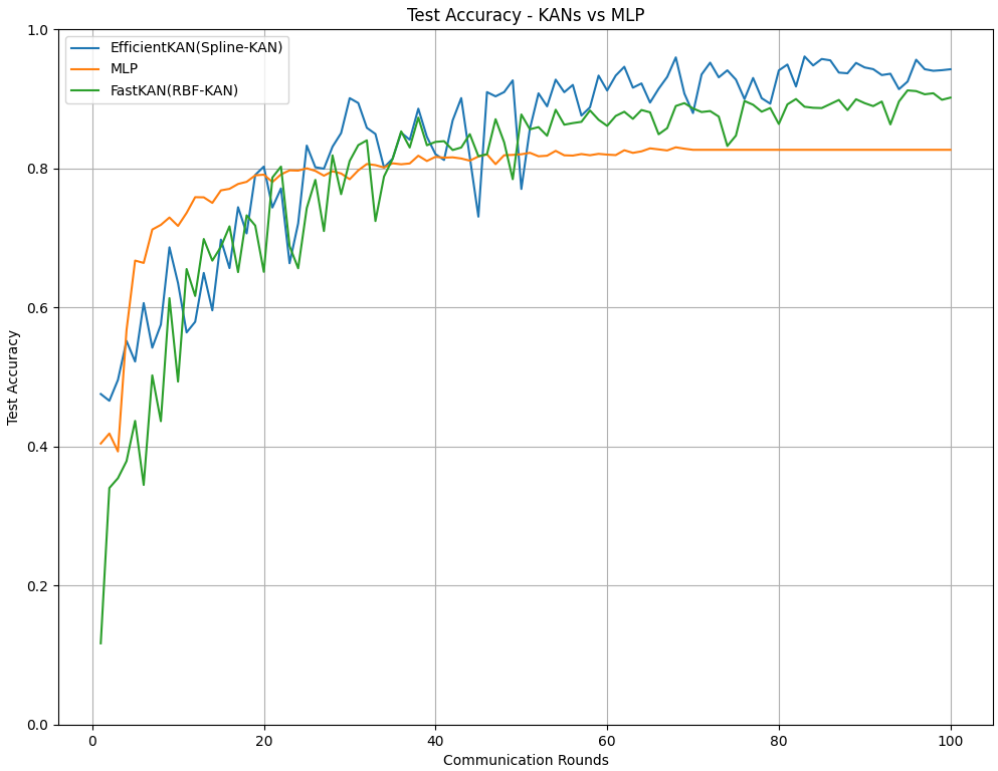

Figure 5: Test Accuracy.

B. Training and Test Loss Analysis

Training loss curves offer more information about how to optimise a model. MLP's training loss, which exhibits larger shifts but a consistent decline, settles at 0.0071 after 100 rounds. FastKAN's loss stabilises at around 0.010 and declines more smoothly. EfficientKAN exhibits the most notable decrease, with loss values staying consistent and falling below 0.005 after 50 rounds. This rapid loss minimisation demonstrates how well EfficientKAN optimises, enabling faster convergence in resource-constrained federated contexts.

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

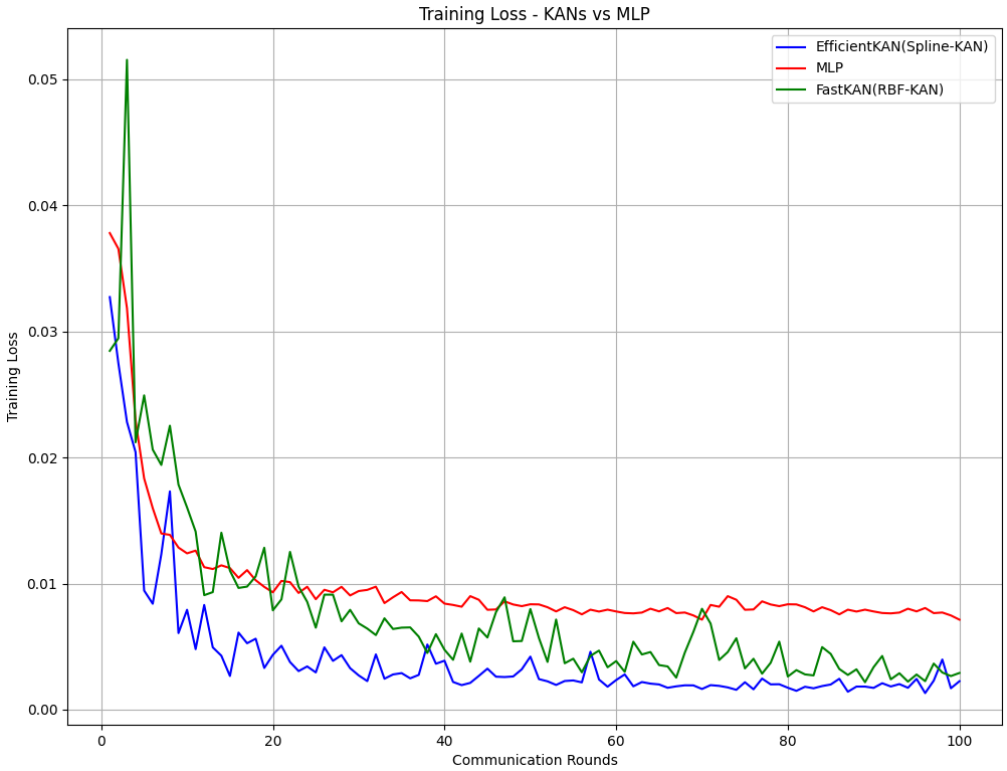

Figure 6: Train Loss.

The test loss analysis in Figure 7 shows the accuracy results. Less than ideal generalisation is suggested by the test loss for MLP, which stabilises at around 0.020. FastKAN reduces test loss to roughly 0.012, which enhances generalisation. EfficientKAN consistently achieves the lowest test loss of roughly 0.008 in most cases. The consistent low test loss confirms EfficientKAN's ability to generalise effectively while lowering prediction errors, which is a significant advantage in federated learning applications.

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

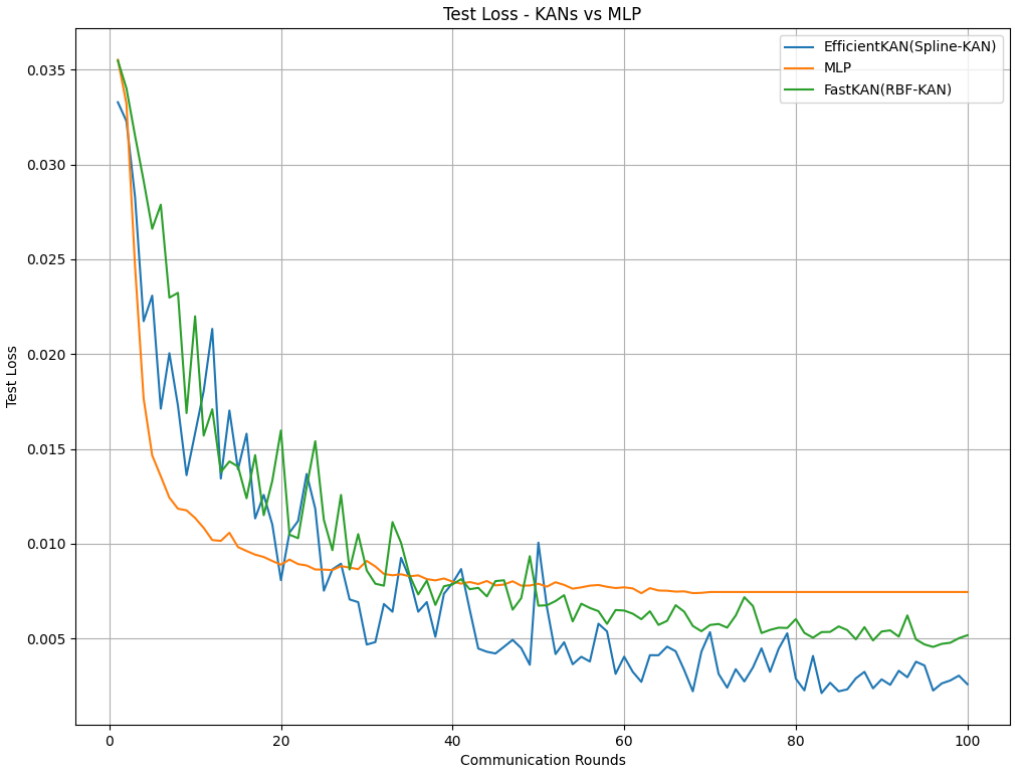

Figure 7: Test Loss.

C. Training and Testing Time Analysis

When assessing machine learning models for practical applications, computational efficiency is a crucial consideration. This section with the help of Table looks at the computational resources needed by each model throughout the training and inference stages, whereas the prior metrics concentrated on prediction performance. Understanding these time requirements provides crucial information about the realistic deploy ability of MLP, FastKAN, and EfficientKAN models, especially when considering resource constraints or time-sensitive applications. The research that follows compares training times and inference speeds across all three designs to highlight the trade-offs between model complexity, computational requirements, and performance advantages.

Table 5: Execution time analysis for different models over 100 rounds

Model	Training Time(seconds)	Testing Time(seconds)
MLP	4192.04	872.31
EfficientKAN	6387.61	1396.25
FastKAN	5474.83	1208.77

Training duration analysis reveals trade-offs between computational efficiency and performance. As shown in Figure 8, training takes approximately 4192.04 seconds for MLP, 5474.83 seconds for FastKAN, and 6387.61 seconds for EfficientKAN, which takes the longest. Particularly when accuracy is more crucial than speed, EfficientKAN's superior performance justifies its higher computing cost.

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

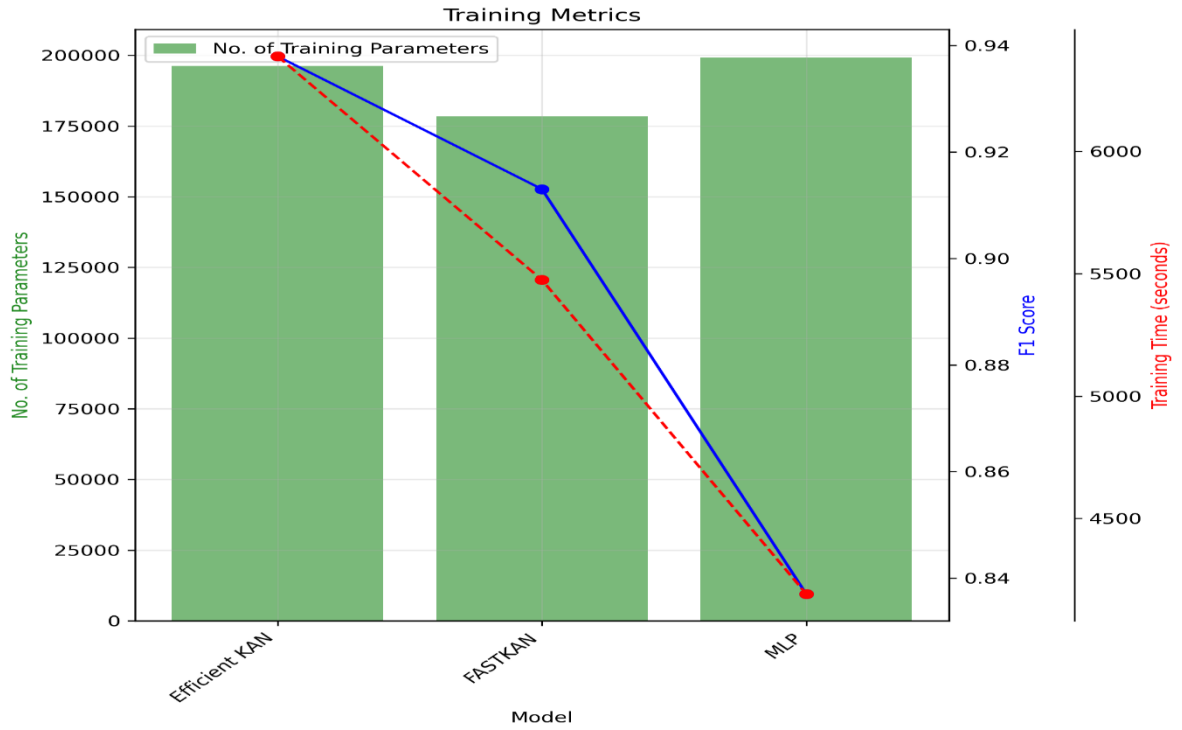

Figure 8: Relationship between F1 training score, number of training parameters, and training time across FashionMNIST dataset.

Similar trends can be seen in testing time, with MLP finishing tests in about 872.31 seconds, FastKAN in about 1208.77 seconds, and EfficientKAN in about 1396.25 seconds as illustrated in Figure 9. EfficientKAN's remarkable accuracy and generalisation make it appropriate for situations where precision exceeds latency concerns, despite its lengthier testing time appearing prohibitive.

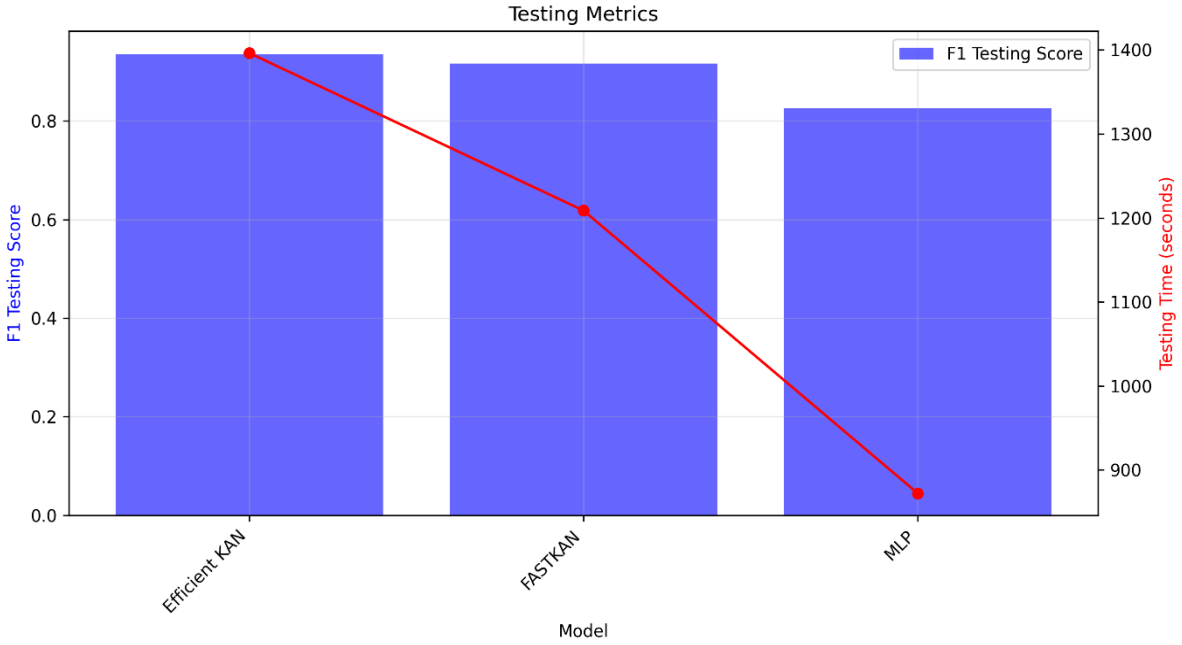

Figure 9: F1 testing score and testing time on FashionMNIST dataset.

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

D. Overall Metrics

To further describe the comparison of these models, we use Figure 10 that allows us to compare the performance of each model on previously trained data with new, untested data.

Out of the three models, the Multilayer Perceptron (MLP) model performs the worst. It achieves about 83.6% accuracy, 82.4% precision, 81.9% recall, and 82.1% F1 score on the training data. The exam data shows a modest decline in performance, with 81.2% F1 score, 80.3% recall, 82.2% precision, and 82.6% accuracy. This slight decline between training and test performance indicates that the model isn't overfitting considerably, but its overall performance is low compared to the other models.

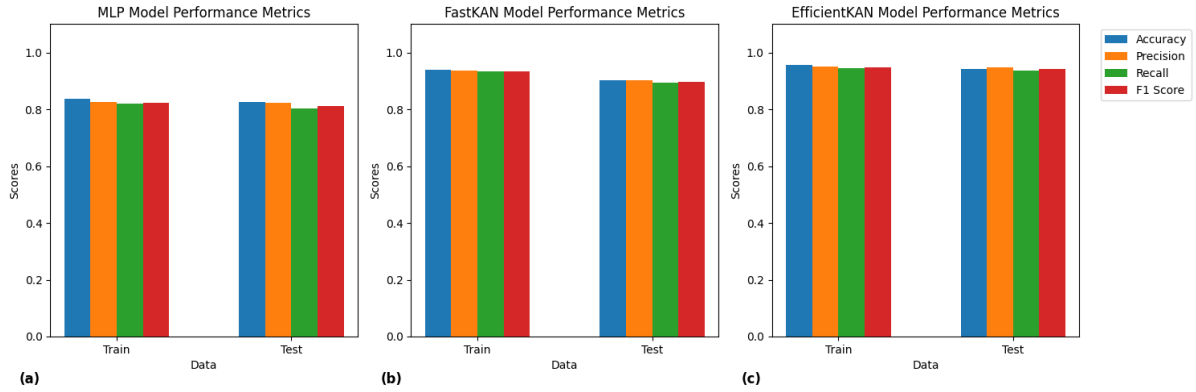

Figure 10: Performance accuracy of three models on FashionMNIST dataset.

The FastKAN model outperforms the MLP model by a significant margin. On training data, it achieves roughly 93.7% accuracy, 93.5% precision, 93.3% recall, and 93.4% F1 score. On test data, performance slightly deteriorates to 90.1% accuracy, 89.7% precision, 90.1% recall, and 89.5% F1 score. Even though the difference between training and test performance is more noticeable (roughly 4 percentage points), the test metrics are still significantly better than the MLP model, suggesting that FastKAN has superior predictive power. The EfficientKAN model outperforms the other two models. On training data, it attains approximately 95.4% accuracy, 95.0% precision, 94.5% recall, and 94.7% F1 score. With 94.2% accuracy, 94.7% precision, 93.5% recall, and 94.1% F1 score on test data, it keeps up its strong performance. The small difference between training and test measures (roughly 1%) suggests excellent generalisation ability, demonstrating the model's strength and balance against overfitting. EfficientKAN appears to be the best model among the three since it continuously maintains high performance metrics, even on test data. This implies that it would be the most dependable option for practical uses where the model will come into contact with fresh, untested data.

CONCLUSION

Practically, this work delivers deployable solutions for federated KANs (F-KANs), including an optimized framework with EfficientKAN and FastKAN variants that bridge theory-practice gaps, architecture selection guidelines for heterogeneous environments, and specialized optimizations (KAN-specific aggregation, adaptive learning rates) accelerating convergence. We further establish a standardized benchmark for evaluating F-KANs under privacy/data heterogeneity scenarios. However, limitations existence like Validation used only FashionMNIST a less complex dataset where performance gaps may not reflect real-world applications also Privacy analysis was restricted to DP-FedPer, while other methods such., homomorphic encryption may yield different outcomes and experiments omitted real-edge challenges like device heterogeneity, and also benefits for non-classification tasks such as reinforcement learning remain unexplored. Future work will thus prioritize architectural innovations Integration with complementary privacy techniques, validation on complex datasets/real-edge deployments and privacy-aware KAN training strategies.

iJETRM
International Journal of Engineering Technology Research & Management
Published By:
https://ijetrm.com/

REFERENCES

[1] Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., & Gao, Y. (2021). A survey on federated learning. *Knowledge-Based Systems*, *216*, 106775.

[2] Zhu, H., Xu, J., Liu, S., & Jin, Y. (2021). Federated learning on non-IID data: A survey. Neurocomputing, 465, 371-390.

[3] Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., ... & Tegmark, M. (2024). Kan: Kolmogorov-arnold networks. arXiv preprint arXiv:2404.19756.

[4] Bodner, A. D., Tepsich, A. S., Spolski, J. N., & Pourteau, S. (2024). Convolutional kolmogorov-arnold networks. arXiv preprint arXiv:2406.13155.

[5] Vaca-Rubio, C. J., Blanco, L., Pereira, R., & Caus, M. (2024). Kolmogorov-arnold networks (kans) for time series analysis. arXiv preprint arXiv:2405.08790.

[6] Zeydan, E., Vaca-Rubio, C. J., Blanco, L., Pereira, R., Caus, M., & Aydeger, A. (2024). F-KANs: Federated Kolmogorov-Arnold Networks. arXiv e-prints, arXiv-2407.

[7] Zeydan, E., Vaca-Rubio, C. J., Blanco, L., Pereira, R., Caus, M., & Dev, K. (2025). Fed-KAN: Federated Learning with Kolmogorov-Arnold Networks for Traffic Prediction. arXiv preprint arXiv:2503.00154.

[8] Sasse, A. M., & de Farias, C. M. (2024). Evaluating Federated Kolmogorov-Arnold Networks on Non-IID Data. arXiv preprint arXiv:2410.08961.

[9] Amouri, A., Al Rahhal, M. M., Bazi, Y., Butun, I., & Mahgoub, I. (2024, October). Enhancing Intrusion Detection in IoT Environments: An Advanced Ensemble Approach Using Kolmogorov-Arnold Networks. In 2024 International Symposium on Networks, Computers and Communications (ISNCC) (pp. 1-6). IEEE.

[10] Zhou, Q., Pei, C., Sun, F., Han, J., Gao, Z., Pei, D., ... & Li, J. (2024). KAN-AD: Time series anomaly detection with Kolmogorov-Arnold networks. arXiv preprint arXiv:2411.00278.

[11] Ta, H. T. (2024, December). BSRBF-KAN: a combination of B-splines and radial basis functions in Kolmogorov-Arnold networks. In International Symposium on Information and Communication Technology (pp. 3-15). Singapore: Springer Nature Singapore.

[12] Li, Z. (2024). Kolmogorov-arnold networks are radial basis function networks. arXiv preprint arXiv:2405.06721.

[13] [113] Basina, D., Vishal, J. R., Choudhary, A., & Chakravarthi, B. (2024). KAT to KANs: A Review of Kolmogorov-Arnold Networks and the Neural Leap Forward. arXiv preprint arXiv:2411.10622

[14] Arivazhagan, M. G., Aggarwal, V., Singh, A. K., & Choudhary, S. (2019). Federated learning with personalization layers. arXiv preprint arXiv:1912.00818.